МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 50.00.00.000 ПЗ

Група ІПм-22-5

Михайлов Тарас

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Михайлов Тарас Олександрович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та алгоритми оцінки та прогнозування виконуваності задач в програмній інженерії

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Михайлов Т.О.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Зікратий Сергій Вікторович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

В.о. завідувача кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
В.о. зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2023 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Михайлову Тарасу Олександровичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “Моделі, методи та алгоритми прогнозування виконуваності задач в програмній інженерії**”**
керівник проекту (роботи) Зікратий Сергій Вікторович, к.т.н., доцент
затверджені наказом закладу вищої освіти від **“** 09 ” листопада 2022 р. № 561/7
2. Строк подання студентом проекту (роботи) 15 січня 2024 р.
3. Вихідні дані до проекту (роботи) Архітектура, формальний опис та алгоритми прогнозування виконуваності задач
4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)
1. Аналіз підходів до рішення задачі “оцінки та прогнозування виконуваності задач в програмній інженерії”
2. Дослідження існуючих моделей, методів та алгоритмів прогнозування виконуваності задач в програмній інженерії
3. Практичне застосування підходу оцінки та прогнозування виконуваності задач для IT проекту
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
1. Діапазон помилок оцінки під час проектів Баррі Боема (рис. 1.1, ст. 21)
2. Оцінювання проекту за допомогою техніки PERT (рис. 3.1, ст. 83)
3. Оцінювання проекту за допомогою техніки PERT із додаванням найбільш ймовірного сценарію (рис. 3.2, ст. 85)
4. Оцінювання проекту за допомогою техніки PERT із додаванням найбільш очікуваного сценарію (рис. 3.3, ст. 86)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Нормоконтроль	доц. к.т.н. Вовк Р. Б.	
Перевірка на плагіат	доц. к.т.н. Вовк Р. Б.	

7. Дата видачі завдання 04 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	20.09.2023	виконано
2	Аналіз сучасних підходів прогнозування виконуваності задач	01.10.2023	виконано
3	Аналіз існуючих моделей, методів та алгоритмів оцінки та прогнозування виконуваності задач	20.10.2023	виконано
4	Дослідження техніки експертної оцінки	15.11.2023	виконано
5	Розрахунок діапазонів оцінки за допомогою техніки експертної оцінки на практиці	03.12.2023	виконано
6	Затвердження пояснювальної записки роботи завідувачем кафедри	15.01.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 99 с., 4 рис., 76 джерел.

Тема: Моделі, методи та алгоритми оцінки та прогнозування виконуваності задач в програмній інженерії.

Об'єкт дослідження: моделі та алгоритми прогнозування виконуваності задач в програмній інженерії.

Мета роботи: удосконалення алгоритму, розробка моделі та механізму прогнозування виконуваності задач в програмній інженерії.

Предмет дослідження: моделі виконуваності задач, методи оцінки виконуваності, алгоритми прогнозування виконуваності, практичні застосування.

Результати дослідження:

Виконано аналіз існуючих моделей, методів та алгоритмів прогнозування виконуваності задач в програмній інженерії і на їх основі запропоновано власний підхід з розрахунком діапазонів оцінки прогнозування за допомогою техніки експертної оцінки.

Висновок:

В результаті досліджень було запропоновано власний підхід з розрахунком діапазонів оцінки прогнозування за допомогою техніки експертної оцінки та втілено його на практиці.

КЕРУВАННЯ ПРОЕКТАМИ, ОЦІНКА ЗАЧОЗАТРАТНОСТІ ПРОЕКТУ, ОЦІНКА ВИКОНУВАНОСТІ ЗАДАЧ, PERT ТЕХНІКА, ЕКСПЕРТНА ОЦІНКА, ОДНОБАЛЬНА ОЦІНКА, ДОВІРЧІ ІНТЕРВАЛИ.

ANNOTATION

Master's work: 99 p., 4 fig., 76 sources.

Topic: Models, methods and algorithms for estimating and projecting the feasibility of tasks in software engineering.

Object of research: models and algorithms for projecting the feasibility of tasks in software engineering.

Purpose of research: improving the algorithm, developing a model and mechanism for forecasting performance of tasks in software engineering.

Subject of research: feasibility task models, feasibility estimation methods, feasibility projection algorithms, practical applications.

Research results:

Performed an analysis of existing models, methods and algorithms for estimating and projecting the feasibility of tasks in software engineering and on its basis is proposed the own approach with calculating estimation ranges using judgement estimation technique.

Conclusion:

As a result of the research, it was proposed its own approach with calculating estimation ranges using judgement estimation technique and implemented it in practice.

PROJECT MANAGEMENT, PROJECT ESTIMATION, SOFTWARE ESTIMATION, PERT TECHNIQUE, JUDGEMENT ESTIMATION, SINGLE-POINT ESTIMATION, CONFIDENCE INTERVALS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ARIMA - Autoregressive Integrated Moving Average, авторегресивний інтегрований рухомий середній аналіз

CMMI - Capability Maturity Model Integration, інтеграція моделі зрілості можливостей

CPA - Critical Path Analysis, аналіз критичного шляху

FPA - Function Point Analysis, аналіз функціональних точок

LOC - Lines of Code, рядкі коду

LOOCV - Leave-One-Out Cross-Validation, перехресна валідація з вилученням одного об'єкта

ML - Machine Learning, машинне навчання

MPE - Mean Percentage Error, середня відсоткова похибка

MRE - Mean Relative Error, середня відносна похибка

PERT - Program Evaluation and Review Technique, техніка оцінки та Перегляду програми

PMBOK - Project Management Body of Knowledge, сукупність знань з управління проектами

R^2 - Coefficient of Determination, коефіцієнт визначення

RMSE - Root Mean Squared Error, корінь середньоквадратичної похибки

ROC - Receiver Operating Characteristic, робоча характеристика приймача

WBS - Work Breakdown Structure, структура декомпозиції робіт

WIP - Work in Progress, робота в процесі

ВСТУП

Актуальність роботи

В динамічному середовищі інженерії програмного забезпечення точна оцінка та прогнозування можливості виконання завдань відіграють ключову роль у успіху проекту [1]. Оскільки ця галузь продовжує розвиватися і пристосовуватися до зростаючих викликів, попит на стабільні моделі, методи та алгоритми для оцінки та прогнозування можливості виконання завдань зросло в кількісному вираженні. Дана робота розпочинає дослідження в галузі інженерії програмного забезпечення з метою вивчення, розробки та вдосконалення технік, які ефективно можуть вирішувати ці критичні аспекти управління проектами.

Галузь інженерії програмного забезпечення сильно покладається на точну оцінку можливості виконання завдань, оскільки це допомагає бізнесу зрозуміти, коли може бути розроблено функціональність та скільки це буде коштувати. Так само команди розробки можуть використовувати оцінку для прогнозування скільки роботи можна виконати в найближчий період часу і допомагати бізнесу управляти розробкою функціоналу.

Дослідження проблеми оцінки програмного забезпечення вимагає аналізу таких основних концепцій, як природа самого програмного забезпечення [2-6], тонкощі управління проектами [7-12], динаміка команд розробки [13-18], управління невизначеністю та ризиками [19-26], історичні дані та метрики [27-30]. Ці фундаментальні елементи становлять основу нашого розуміння та підходу до оцінки програмного забезпечення.

Програмне забезпечення, будучи невловимою та високоманевренною сутністю, представляє унікальні виклики для оцінки. Воно не має фізичних властивостей, які роблять оцінку прямолінійною в таких галузях як будівництво. Замість цього програмне забезпечення характеризується складністю, змінюваністю та зростаючими вимогами. Таким чином, розуміння суті програмного забезпечення та того, як воно еволюціонує під час розробки, є основною частиною оцінки [31].

Ефективна оцінка програмного забезпечення нерозривно пов'язана із практиками управління проектами. Керівники проектів повинні управляти ресурсами, графіками та бюджетами, зберігаючи делікатний баланс між обсягом проекту та обмеженнями. Розуміння принципів та методологій управління проектами є важливим для вирівнювання зусиль з оцінкою до цілей проекту [11].

Оцінка програмного забезпечення не відбувається в ізоляції; це спільна діяльність. Команди розробки з великою кількістю ролей та навичок взаємодіють, щоб привести проект до успішного завершення. Визнання динаміки цих команд, їхніх можливостей та шаблонів комунікації є вирішальним для точної оцінки [32].

Оцінка в інженерії програмного забезпечення не може уникнути спектру невизначеності та ризиків. Проекти піддаються непередбачуваним викликам, змінюваним вимогам та зовнішнім факторам. Комплексна структура оцінки повинна включати в себе оцінку ризиків та стратегії їх зменшення для врахування властивих невизначеностей в розробці програмного забезпечення [33].

Використання історичних даних та відповідних метрик – ще один важливий аспект оцінки програмного забезпечення. Аналіз минулих проектів може дати уявлення про закономірності, тенденції та ефективність. Цей підхід на основі даних допомагає вдосконалювати моделі оцінки та підвищувати точність з часом [34].

Область оцінки програмного забезпечення насичена різноманітним спектром інструментів та технік. Це охоплює все від експертної думки та аналогічної оцінки до більш високорозвинених підходів, таких як параметричне моделювання та машинне навчання. Ознайомлення з цими інструментами та їх застосованість до конкретних сценаріїв проекту є базовим [35].

По суті, проблема оцінки програмного забезпечення – це багатогранний виклик, який виходить за межі простого числового аспекту. Це дисципліна, яка вимагає цілісного сприйняття природи програмного забезпечення, ефективних практик управління проектами, динаміки співпраці командрозробки, оцінки ризиків, аналізу історичних даних та універсального набору методологій оцінки.

У багатьох випадках бажання отримати точну оцінку може призвести до "гри з числами", коли кожна функція завищується або “обрізаються кути”, щоб

відповідати оцінкам. Це може призвести до непрозорого характеру роботи і втрати цінності. Незважаючи на ці труднощі, все ще існує потреба у мінімізації рівня припущень у оцінках та покращенні точності прогнозів.

Зв'язок роботи з науковими програмами, планами, темами

Це дослідження відповідає сучасним планам та програмам галузі, зосереджуючись на критичних аспектах можливості інженерії програмного забезпечення. Воно безпосередньо сприяє актуальним науковим темам, наголошуючи на практичній актуальності та застосовності запропонованих моделей, методів та алгоритмів для вирішення потреб галузі.

Мета і задачі дослідження

Метою даної магістерської роботи є розробка набору моделей, методів та алгоритмів, які можна використовувати для оцінки та прогнозування можливості виконання завдань в інженерії програмного забезпечення. Крім того, крім пропозиції нових моделей та методів, ця робота також спрямована на вирішення проблем, пов'язаних із неточністю оцінки, представляючи набір алгоритмів та технік для виявлення критичних завдань у проектах розробки програмного забезпечення. Ці інструменти можуть допомогти командам розробки покращити свої процеси планування та виконання проекту, мінімізуючи рівень припущень, втіленого в оцінках.

Додатковою ідеєю даної роботи є демонстрація того, як працювати з оцінками після виконання оціненої роботи. Ця частина зазвичай пропускається, однак вона є дуже важливою, оскільки дозволяє розрахувати Mean Relative Error (MRE). MRE дозволяє вимірювати точність наших оцінок та, в кінцевому підсумку, використовувати ці дані як можливе відхилення у майбутніх оцінках.

Об'єктом дослідження даної магістерської роботи є розробка моделей та алгоритмів для прогнозування можливості виконання завдань в інженерії програмного забезпечення.

Предметом дослідження даної магістерської роботи є моделі виконуваності задач, методи оцінки виконуваності, алгоритми прогнозування виконуваності, практичні застосування.

Методи дослідження. Були використані наступні методи оцінки програмного забезпечення у реальному проекті:

- Методика експертної оцінки
- Методика однобальної оцінки
- PERT
- Порівняння оцінок з фактичними результатами

Наукова новизна одержаних результатів. Окрім використання на реальному проекті, було введено "порівняння оцінок із фактичними результатами". Внаслідок цього порівняння був запропонований Mean Relative Error (MRE) коефіцієнт, який в кінцевому підсумку використовувався для всіх подальших оцінок.

Практичне значення одержаних результатів. Команда провела кілька ітерацій оцінки прогнозування виконання завдання, використовуючи різні техніки. Насамперед, це допомогло більш точно оцінити це завдання. Крім цього, було виведено Mean Relative Error (MRE) коефіцієнт, що дав чітке розуміння похибки оцінки командою. Цей коефіцієнт допоміг візуалізувати рівень похибки конкретною командою при прогнозуванні виконання завдань, що в свою чергу має допомогти більш точно проводити такі оцінки.

Особистий внесок. Проведення реальних експериментів на реальному проекті та представлення результатів у поточній дисертації.

Публікації. На основі результатів наукового дослідження, проведеного у магістерській роботі, була підготовлена та опублікована наступна стаття “Застосування методів експертної оцінки при прогнозуванні термінів виконання проектів в галузі ІТ.” в журналі “Інформаційні технології та суспільство” Міжрегіональної Академії управління персоналом, № 2 (8) (2023).

Структура магістерської роботи. Магістерська робота представлена на 99 сторінках друкованого тексту, який включає в себе вступ, три розділи, висновки та список використаних джерел (76 найменувань).

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО РІШЕННЯ ЗАДАЧІ “ОЦІНКИ ТА ПРОГНОЗУВАННЯ ВИКОНУВАНОСТІ ЗАДАЧ В ПРОГРАМНІЙ ІНЖЕНЕРІЇ”

1.1 Визначення та історичний аналіз оцінки програмного забезпечення

1.1.1 Визначення оцінки програмного забезпечення

Оцінка програмного забезпечення є критичною складовою управління проектами та необхідною для успішного завершення проектів розробки програмного забезпечення [35]. У своїй основі оцінка - це прогноз того, як довго триватиме проект, скільки він буде коштувати та скільки ресурсів йому необхідно. Однак важливо відзначити, що оцінка не є гарантією або зобов'язанням. Це швидше краще вгадування на основі інформації, доступної на момент складання оцінки.

Оцінку слід розглядати як діапазон можливостей, а не як одне число. Цей діапазон слід повідомляти зацікавленим сторонам разом із припущеннями та ризиками, пов'язаними з оцінкою. Чим більше інформації доступно проекту, тим меньшим буде діапазон оцінки.

1.1.2 Історичний аналіз оцінки програмного забезпечення

На початковому етапі розвитку програмного забезпечення оцінка була новоствореним поняттям, часто затемненим новизною створення програм для комп'ютерів, що тільки зароджувались. Перші оцінювачі стикалися з відсутністю встановлених методологій, історичних даних та чіткого розуміння нюансів, пов'язаних з передбаченням термінів та потреб у ресурсах проекту.

З розвитком програмних проектів модель Waterfall виросла в домінуючу методологію. Оцінка в цей період слідувала послідовному підходу, при якому кожна фаза будувалася на попередній. Хоча цей метод надавав структуру розробці, жорстка природа послідовної оцінки виявилася важкою для врахування змінних вимог проекту.

Прихід методологій Agile викликав зміну парадигми в розробці програмного забезпечення, маючи глибокий вплив на практики оцінки. Ітеративна та інкрементальна розробка внесла більш гнучкий підхід, дозволяючи оцінювачам адаптуватися до змінних основ проекту. Agile-оцінка будувалась на співпраці, зворотному зв'язку та постійній зміні, відображаючи відхід від традиційних, фіксованих моделей.

З ростом складності та розмірів програмних проектів оцінка стикнулася з новими викликами. Тонкощі великих систем, поєднані з взаємопов'язаним характером сучасного програмного забезпечення, вимагали більш витонченого підходу. Оцінювачі боролися з складнощами точного передбачення зусиль, часу та ресурсів, необхідних для заплутаних та багатогранних проектів.

З часом індустрія програмного забезпечення визнала потребу в стандартизованих практиках оцінки. Організації та професіонали почали впроваджувати передові практики, такі як Capability Maturity Model Integration (CMMI) та Project Management Body of Knowledge (PMBOK), надаючи фреймворки для більш систематичних та надійних процесів оцінки.

Інтеграція статистичних моделей та метрик стала відзначенням високорозвинутих практик оцінки. Концепції, такі як Function Points, Lines of Code та Cyclomatic Complexity, стали вимірювальними метриками, що дозволяють оцінювачам застосовувати статистичні моделі для більш точних передбачень.

Історія оцінки програмного забезпечення визначена як успіхами, так і невдачами. Визначні невдачі проектів підкреслили важливість реалістичних оцінок, ефективного управління ризиками та необхідності постійного вдосконалення. Успішні історії надали інсайти про методології, які витримали перевірку часом, наголошуючи на ітеративному та спільному характері сучасної розробки програмного забезпечення.

У сучасній основі оцінки програмного забезпечення виникають труднощі, що виникають від швидких технологічних досягнень, розподілених команд розробників та змінюваних методологій проектів. Оцінювачі вивчають новаторські підходи,

використовуючи штучний інтелект, машинне навчання та автоматизацію для підвищення точності та адаптивності моделей оцінки.

Подорож оцінки програмного забезпечення далека від завершення. Вона продовжує розвиватися, формуючись на вивчених уроках минулого та відповідаючи вимогам майбутнього. Оцінювачі сьогодні стоять на плечах тих, хто пройшов труднощі попередніх епох, збагачені різноманіттям досвіду, який сприяє постійному удосконаленню практик оцінки.

1.1.3 Еволюція технік оцінювання

Еволюція технік оцінювання в сфері інженерії програмного забезпечення відображає трансформаційну подорож крізь літопис управління проектами. Від простих початків до вишуканих методологій сучасності, основи оцінки та прогнозування можливості виконання завдань у розробці програмного забезпечення пережили неймовірну еволюцію.

На початкових етапах інженерії програмного забезпечення оцінка визначалася простотою. Ранні виконавці використовували базові моделі та евристичні методи для передбачення термінів проекту, витрат та потреб у ресурсах. Ці початкові техніки, надаючи початкові уявлення, поклали основу для наступних досягнень.

З розвитком розробки програмного забезпечення змінювалася і концепція оцінювання. Парадигмальні зміни визначали трансформаційні моменти, такі як прийняття ітеративного розвитку та відхід від жорстких лінійних методологій. Визнання того, що програмні проекти мають вроджено динамічний та ітеративний характер, спонукало до переходу до більш спільного та адаптивного підходу до оцінювання.

Рішучий етап у еволюції оцінювання пов'язаний з появою кількісних моделей. Вийшовши за межі суб'єктивного експертного судження, галузь взяла за основу статистичні та математичні моделі. Параметричні моделі оцінювання та включення емпіричних даних стали необхідними, покращуючи точність оцінок та впроваджуючи підхід на основі даних.

У сучасній ері продвинуті статистичні методи та машинне навчання відкрили новий підхід для технік оцінювання. Штучний інтелект, разом з обширними наборами даних, дозволяє досягти рівня точності та адаптивності, які раніше були невидимими. Алгоритми тепер можуть аналізувати історичні дані проектів, визначати закономірності та робити обґрунтовані прогнози, пропонуючи більш удосконалений підхід до оцінювання.

Однак ця еволюційна траєкторія не пройшла без викликів. Кожен етап розвитку супроводжується труднощами, невдачами та цінними уроками. Динаміка програмних проектів, спільно з технологіями що розвиваються, вимагає постійного вдосконалення та адаптації технік оцінювання. Вивчення минулих викликів є важливою складовою навігації складнощами постійно змінних основ розробки програмного забезпечення.

Еволюція технік оцінювання в інженерії програмного забезпечення відображає постійне прагнення до точності та адаптивності. Від скромних початків галузь прийняла складність, впроваджуючи високорозвинені моделі та технології для задоволення вимог сучасної розробки програмного забезпечення. Розуміння цієї еволюції є важливим для практиків, які намагаються орієнтуватися в тонкощах оцінювання та прогнозування можливості виконання завдань у постійно змінному світі інженерії програмного забезпечення.

1.2 Графіки розробки програмного забезпечення

1.2.1 Мета графіків

Графіки мають три основні функції, незалежно від того чи це організація соціальної події чи оновлення веб-сайту [36]. Перша мета полягає в установленні зобов'язань щодо того, коли завдання будуть виконані, причому графік слугує обов’язковою угодою між усіма учасниками щодо того, що кожна людина виконає протягом визначеного часу. Зазвичай люди асоціюють проектні графіки з цією першою метою, яка спрямована на зовнішніх зацікавлених сторін, а не на команду проекту. Графіки часто використовуються для допомоги укладення контрактів чи

виконання термінів клієнта, причому замовники часто сплачують як за послугу, так і за графік. Таким чином, необхідно встановити взаємопогоджений час для виконання конкретних завдань, щоб замовники чи партнери могли робити плани на основі прогресу проекту.

Друга мета графіка - підштовхувати людей сприймати свій внесок як частину більшої системи та зобов'язуватися робити свою роботу відповідно до інших. Без попереднього графіка, що визначає конкретні терміни та строки, ймовірно, не буде виявлено зв'язків та взаємозалежностей. Без графіка люди зазвичай концентруються лише на своїх власних завданнях і ігнорують вплив своєї роботи на інших.

Тільки тоді, коли деталі документовані із зазначенням імені людей поруч, можливі фактичні оцінки та аналіз припущень. Це стосується навіть невеликих команд або індивідуумів, які працюють самостійно. Графік має психологічну силу, оскільки він публікує зобов'язання, які приймаються. Як тільки він розміщений на дошці в коридорі, він слугує постійним нагадуванням команді про те, що потрібно виконати, що робить складнішим забути чи пропустити щось. Крім того, зокрема для менеджерів проекту, попередній графік служить рамкою для перевірки можливості виконання певних завдань та порівняння обсягу проекту з тим, що практично досяжно.

Функція примусу вказує на психологічний зсув, який призводить до зміни перспективи, ставлення чи поведінки. Графіки служать важливими функціями виклику для проектів, оскільки вони змушують всіх, хто взяв участь, ретельно обдумати роботу, яку потрібно виконати. Це важливий крок у реалізації потенціалу проекту. Навіть якщо графік не цілком точний і запізнюється, подвоюється чи скорочується, зобов'язання та зв'язки, встановлені під час складання графіка, все одно можна утримувати. Таким чином, друга мета графіка може бути досягнута і бути цінною, навіть якщо сам графік виявиться дуже неточним. Наприклад, якщо проект завершується набагато пізніше, ніж планувалося, наявність графіка все одно може сприяти завершенню проекту.

Графіки служать третьою метою, яка полягає в розбитті роботи на управляючі фрагменти та наданні інструмента для відстеження прогресу. Коли робота

розбивається на менші частини, людям легше розуміти, що потрібно робити. Наприклад, якщо будівельник надає лише одну цифру, вказуючи "Будинок: 120 днів", важко зрозуміти роботу, яку потрібно виконати. Однак, якщо будівельник надає тиждень-за-тижнем розбивку дій, кожен може розуміти, які завдання будуть виконані, коли вони будуть виконані і які є пріоритети. Такий рівень деталізації дозволяє ставити конкретні запитання та уточнювати припущення. Добрий графік також надає менеджеру проекту чіткий погляд на проект, допомагає виявляти труднощі та прогалини на ранніх етапах та збільшує шанси на успіх.

Графіки стають все важливішими зі збільшенням розміру та складності проектів. З більшою кількістю учасників з'являється більше взаємозалежностей і більше шансів, що рішення та час можуть впливати на інших. З іншого боку, в невеликій команді з кількома людьми ймовірніше, що вони визначать проблеми в роботі один одного.

У невеликих командах, поки відставання від графіку все ще небажане, його можна легше виправити. Наприклад, запізнення на півдня впливає лише на кількох людей і може бути компенсоване додатковими годинами роботи або залученням допомоги від решти команди. На відміну від цього, на великих проектах з десятками або сотнями людей та компонентів, навіть одноденне запізнення може мати каскадні ефекти і призводити до непередбачуваних проблем, важких до вирішення. Тим не менш, незалежно від розміру команди, наявність графіка дозволяє менеджерам та тим, хто відповідає за фінансове управління, ставити питання, робити необхідні корективи та підтримувати команду, активно виявляючи та вирішуючи проблеми по мірі їх виникнення.

Розглядаючи ці три мети, стає очевидним, що графіки не є панацеєю для всіх проблем проекту. Незважаючи на час і зусилля, витрачені на створення графіків, вони не можуть виправити конструкційні чи інженерні недоліки або компенсувати нестачу лідерства, невизначені цілі чи неприйнятну комунікацію. Таким чином, графіки - це лише набір цифр і термінів, і на індивідах лежить відповідальність використовувати їх як інструмент для управління та підтримки руху проекту вперед.

1.2.2 Причини невдачі графіків

Графіки проектів часто несуть основне тягар вини за будь-які непорозуміння, що виникають [37]. Чи то неправильна оцінка, пропущені вимоги чи непередбачений інцидент, графік і людина, відповідальна за нього, зазвичай несуть відповідальність. Навіть якщо сталася перебій в електропостачанні, який триває 10 днів, чи найкращі програмісти команди захворіли, хтось вказуватиме на графік як на причину затримки і обтяжуватиме вину менеджера, відповідального за графік. Це несправедливо, але це часто трапляється. Незважаючи на нелюбов, графіки все ще тримаються недосяжного стандарту. Навіть найкращі графіки, які оснащені найбільш удосконаленими інструментами, мають труднощі передбачення майбутнього, що не є сильною стороною нашого виду.

Однак команда, яка визнає загальні причини відставань від графіка і вживає заходів для зменшення цих ризиків, може створити більш корисний і точний графік, який підтримує процес розробки.

Одна з основних проблем полягає в тому, що на початковому етапі створення графіка є багато невідомого. Коли графік формується під час початкової фази планування, ще не прийнято безліч рішень, які можуть вплинути на графік. Виникають виклики та непередбачені проблеми, які ранній план не може передбачити. Менеджер проекту не має достатньо інформації для реалістичних передбачень, доки вимоги не будуть зрозумілими, а високорівневий дизайн не буде вже у роботі. Тим не менш, попередній графік часто створюється за допомогою випадкових цифр та необгрунтованих припущень. Люди часто потрапляють у пастку плутанини влучністю з точністю: графік, який виглядає вражаючими конкретними датами (влучність), може не обов'язково бути вірним відображенням реальності. Хоча влучність легко досягти, точність вимагає багато більше зусиль.

Правда в тому, що кожен проект та графік повинні мати стартову точку. Іноді невизначене припущення може бути корисним способом мотивації команди та встановлення деяких початкових меж. Це може започаткувати процес вивчення для розробки графіків та вирішення критичних питань. Однак виключна залежність від неперевіреної та необдуманої узагальненості для створення графіку несе значущі

ризики і, без подальшого розвитку, може призвести до невигідних результатів. Існують вагомі докази, що для будь-кого складно точно передбачити потрібний час на початку проекту.

У своєму есе 1988 року з програмної інженерії Баррі Бем виявив взаємозв'язок між помилками графіка та часом його оцінки в проекті (як показано на рис. 1.1.). Він виявив, що якщо загальні оцінки графіка робляться рано, потенційна розбіжність може становити до 400%. Із зростанням кількості прийнятих рішень під час фази проектування відхилення зменшується, але все ж залишається значним. Тільки під час виконання відхилення графіку стає більш розумним. Але навіть тоді існує ймовірність відхилення на 20% в точності прийняття рішень щодо графіку.

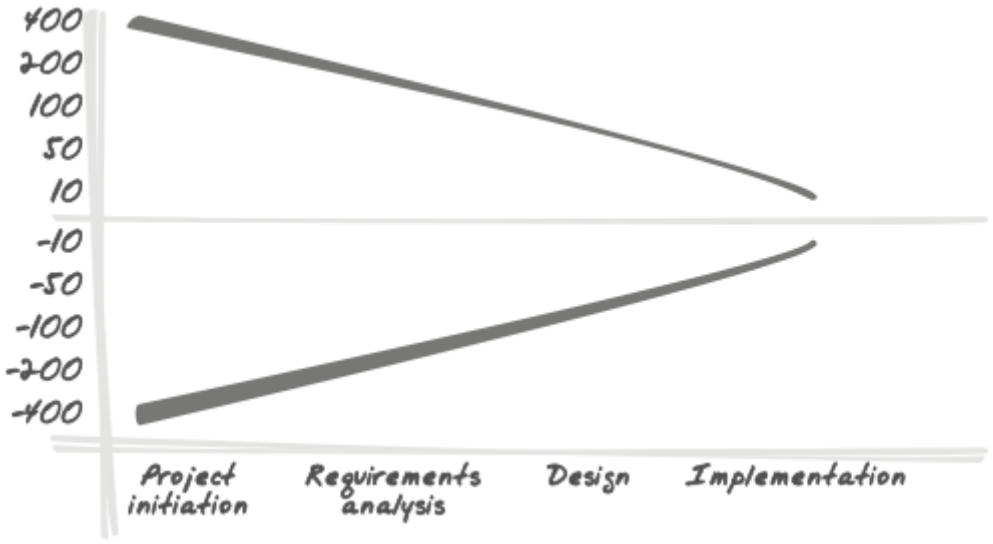

Рис. 1.1. Діапазон помилок оцінки проектів Баррі Боема

Для менеджерів проектів важливо розуміти, що точність оцінки графіка покращується з часом. Це означає, що графіки потребують постійної уваги та коригувань по мірі розвитку проекту.

1.2.3 Графік є набором ймовірностей

Створення ідеальних графіків не можна досягти за допомогою чарівної формули або точної науки. Планування - це загальне завдання, яке враховує багато різних аспектів проекту як у поточний момент, так і в майбутньому. Графіки, по суті, є передбаченням, яке створюється шляхом підсумовування кількох оцінок, кожна з

яких піддається непередбачуваним непорозумінням і проблемам. Ефективні графіки - це результат команди або лідера, які мають сильні знання в різних областях розробки програмного забезпечення. Неможливо стати експертом лише в одній області процесу розробки і в той же час стати професіоналом в графіках.

Якщо команда може домовитися, що графік представляє собою набір ймовірностей, то проблема полягає не в самому графіку, а в його використанні. Коли графік представлений на командному засіданні або розсилках по електронній пошті, важливим питанням буде: наскільки ймовірний вказаний графік? Якщо в графіку немає ймовірностей, таких як п'ять найважливіших ризиків та їх оцінена ймовірність виникнення, і відповідальна особа за створення графіка не може уточнити свої припущення, слід вважати, що графік можливий, але не ймовірний. Команді слід заохочувати пропонувати корективи або модифікації графіка для підвищення ймовірності його успішності.

Головна ідея полягає в тому, що графік не повинен бути ідеальним, враховуючи, що ідеальних графіків не існує. Замість цього графіки повинні бути достатньо хорошими для того, щоб як команда, так і лідери вірили в них, забезпечували основу для моніторингу та корекцій та мали ймовірність досягнення, яка відповідає очікуванням клієнта, компанії чи спонсора проекту в цілому.

1.2.4 Залежність між оцінками та графіками

Оцінки не є тим самим, що й графіки. Оцінки - це лише одна складова процесу планування.

Оцінка - це прогноз того, скільки часу займе проект, скільки він вартуватиме і скільки ресурсів буде потрібно. А от графік - це детальна карта, яка визначає конкретні завдання, дії та важливі пункти, необхідні для завершення проекту. Хоча оцінки є необхідним елементом для створення графіку, вони не представляють собою сам план.

Оцінки завжди можуть змінюватися і повинні оновлюватися протягом життєвого циклу проекту. За мірою того, як стає доступна більше інформації і як

проект розвивається, оцінки можуть бути переглянуті для відображення змін у завданнях, графіках або ресурсах.

Щоб створити ефективний графік, потрібно використовувати оцінку як вихідний пункт, а потім уточнювати її на основі додаткової інформації та аналізу. Цей процес може включати розбиття проекту на менші завдання та оцінювання часу та ресурсів, необхідних для кожного завдання, визначення та зменшення ризиків та врахування відгуків зацікавлених сторін.

Графіки повинні бути гнучкими та пристосованими до змінних обставин. Хоча графік і надає карту для проекту, його не слід вважати закам’янілим. Замість цього його потрібно оновлювати та коригувати за необхідності, щоб бути впевненим, що проект слідує визначеним цілям та завершується успішно.

Точні результати оцінок не можуть бути досягнуті лише за допомогою оціночних практик. Їх потрібно підтримувати ефективним контролем проекту.

Контроль проекту - це процес моніторингу та управління проектом для забезпечення його виконання та досягнення цілей. Цей процес включає в себе моніторинг прогресу, виявлення та зменшення ризиків та внесення змін за необхідності для збереження проекту на правильному шляху.

Оцінки є ключовою складовою контролю проекту, оскільки вони надають базовий рівень для вимірювання прогресу та виявлення потенційних проблем. Однак оцінки є такими ж вдалими, як інформація та припущення, на яких вони базуються. З розвитком проекту та надходженням більше інформації оцінки можуть потребувати перегляду для відображення змін у завданнях, графіках чи ресурсах.

Окрім точних оцінок, ефективний контроль проекту вимагає проактивного та гнучкого підходу до управління проектом. Це включає в себе виявлення та зменшення ризиків якнайскоріше, утримання відкритого спілкування із зацікавленими сторонами та внесення змін за необхідності для збереження проекту на правильному шляху.

Постійно оновлюючи та уточнюючи як оцінки, так і графіком протягом життєвого циклу проекту, проекти розробки програмного забезпечення можуть бути завершені вчасно, в межах бюджету та з виконанням усіх цілей.

1.3 Опис процесу складності виконуваності завдань

1.3.1 Точність оцінювання

В рамках фази проектування, дизайнери, розробники та тестувальники розкладають проект на менші, керовані частини, які можна побудувати. Ці окремі компоненти, часто названі робочими одиницями або Work Breakdown Structure (WBS), подальше стають пунктами, перерахованими в основному графіку проекту. Робочі одиниці (в ідеалі) вдало призначаються команді розробників і, після їх об’єднання, створюється графік. Кожній з цих робочих одиниць розробник повинен призначити конкретний час, тоді розграфік розробляється на основі цих оцінок.

Просто кажучи, точні оцінки роботи вважаються хорошими, тоді як неточні вважаються поганими. Це означає, що керівники команди відповідальні за встановлення стандартів для проекту, що включає активне переглядання та вдосконалення оцінок. Корисно включити команду тестування в процес оцінювання, дозволяючи їм брати участь у обговореннях дизайну та надавати зворотний зв'язок. Це може допомогти їм вдосконалити свої власні оцінки для роботи з тестування, яка може відрізнятися від оцінок роботи програмування. У багатьох випадках у команди з тестування існує унікальна перспектива щодо недоліків дизайну та потенційних випадків невдачі, які інші можуть пропустити.

Більшість людей не любить оцінювати складні завдання, за які вони будуть відповідальні. Поки це може бути цікавим вихвалятися та давати оцінки своїх здібностей, коли закликають поставити чіткі терміни виконання, впевненість пропадає. Зовсім ймовірно, те, до чого ми зобов'язалися сьогодні, може виявитися неможливим або непривабливим, коли настане час. Завдання може бути складнішим, ніж ми очікували. Розробники не відрізняються від інших відчуттям тривоги перед оцінкою з важливих причин. Коли вони вказують певний час для завдання, вони ризикують значними неточностями. Навіть ті розробники, які знають процедуру оцінки і вірять у неї, не люблять її виконувати. Співчуття в цьому плані повинно бути обмеженим, оскільки всі, хто працює в інженерії та будівництві, стикаються з подібними труднощами. Їхні проблеми та методи не здаються фундаментально

відмінними від тих, які зустрічають веб-розробники та інженери програмного забезпечення. Основна відмінність полягає у тому, скільки часу вони мають для створення оцінок та наскільки дисципліновано вони використовують цей час.

1.3.2 Когнітивні упередження

Оцінювання, ключовий аспект управління проектами в інженерії програмного забезпечення, не лише технічне зусилля, але й має глибокий вплив людським когнітивним сприйняттям. Взаємодія між складнощами розробки програмного забезпечення та вродженими упередженнями людського розуміння вносить захопливу міру в процес оцінювання.

Когнітивні упередження - це вроджені короткі шляхи та шаблони мислення, які, хоч і часто корисні для прийняття рішень, можуть призводити до системних відхилень від раціонального судження. У контексті оцінювання ці упередження можуть значно впливати на точність та надійність передбачень. Визнання та розуміння цих упереджень є ключовим для практиків, які мають на меті покращити точність своїх оцінок.

Одним із поширених когнітивних упереджень, що впливає на оцінювання, є упередження переконаності. Керівники проектів та члени команди, піддані природньому бажанню вірити в свої здібності, можуть переоцінювати впевненість у своїх оцінках. Це упередження може призвести до надто оптимістичних передбачень, що потенційно призведе до недооцінки термінів проекту та потреб у ресурсах.

Упередження підтвердження, тенденція схильності до інформації, яка підтверджує існуючі вірування, грає ключову роль в оцінюванні. Члени команди можуть несвідомо шукати дані, які підтримують їхні початкові оцінки, обходячи інформацію, що протирічить. Це упередження може призвести до спотвореного сприйняття можливостей проекту, ускладнюючи виявлення потенційних ризиків та труднощів.

Упередження передбачає занадто сильне покладання на першу зустрічну інформацію при прийнятті рішень. В оцінюванні це упередження може проявлятися

при встановленні початкових параметрів проекту. Після цього внесені коригування можуть бути недостатніми, оскільки оцінювачі прив'язують свої передбачення до початкових даних, що потенційно призводить до неточностей у розподілі ресурсів та термінах проекту.

Ще одним поширеним когнітивним викривленням, що впливає на оцінювання, є упередження оптимізму. Члени команди можуть бути занадто оптимістичними щодо результатів проекту, недооцінюючи ймовірність непередбачених труднощів. Це упередження може сприяти недостатньому плануванню ризиків та запобіжних заходів, що потенційно може завдати ушкоджень проектам у випадку несподіваних труднощів.

Подолання когнітивних упереджень в оцінюванні вимагає активного підходу. Впровадження стратегій, таких як рецензії від колег, різноманітні відгуки від членів команди та використання історичних даних для порівняльного аналізу, може допомогти протидіяти впливу упереджень. Крім того, створення культури відкритого спілкування та прийняття зворотного зв'язку може створити середовище, що сприяє реалістичному оцінюванню.

Визнання вбудованої невизначеності в розробці програмного забезпечення надзвичайно важливе. Замість того, щоб піддаватись упередженням, зумовленим бажанням точності, керівники проектів та оцінювачі можуть скористатися ймовірнісними підходами. Розуміння того, що оцінки представляють собою діапазон можливостей, а не детерміновані значення, дозволяє прийняти більш реалістичний та гнучкий підхід до планування проекту.

У складному світі оцінювання проектів програмного забезпечення, когнітивні упередження показують захопливу наративну лінію впливу людської поведінки на передбачувану точність. Визнання існування упереджень - перший крок у пом'якшенні їхнього впливу. За допомогою культури уважності, прийняття невизначеності та впровадження стратегій протидії упередженням практики можуть впоратися з людським фактором в оцінюванні, підвищуючи надійність передбачень проекту в постійно змінному пейзажі інженерії програмного забезпечення.

1.3.3 Комунікації з зацікавленими особами

У динамічному світі інженерії програмного забезпечення успішне оцінювання проекту - це не лише технічне зусилля; це спільний процес, що тісно пов'язаний з ефективною комунікацією з учасниками процесу.

Чітка комунікація лежить в основі взаєморозуміння між учасниками проекту та командою з оцінювання. Забезпечення того, що всі спільно розуміють бачення проекту та осмислення його цілей, обсягу та обмежень, є надважливим. Учасники, починаючи від клієнтів до внутрішніх членів команди, повинні активно залучатися до діалогу, який сприяє всебічному розумінню основи проекту.

Прозора комунікація - це основа довіри в будь-якому проекті. Коли учасникам проекту надається інформація про процес оцінювання, припущення, які були зроблені, та потенційні ризики, що їх можна ідентифікувати, тоді виникає культура довіри та співпраці. Ця прозорість не лише створює впевненість в результати оцінки, але й сприяє спільній відповідальності за успіх проекту.

Ефективна комунікація з учасниками вирішальна для управління очікуваннями протягом життєвого циклу проекту. Регулярні оновлення та чітке формулювання невизначеностей, притаманні в оцінюванні, допомагають учасникам усвідомити складність ситуації. Проактивне управління очікуваннями запобігає розбіжностям у передбачуваних результатів, мінімізуючи ризик невдоволення з розвитком проекту.

Учасники несуть в собі багатство контекстного знання для процесу оцінювання. Їхні думки, очікування та пріоритети повинні бути активно враховані та враховані в моделі оцінювання. Залучаючи учасників у процес прийняття рішень, оцінювачі отримують всебічне розуміння вимог проекту, забезпечуючи відповідність кінцевих оцінок очікуванням учасників.

У розробці програмного забезпечення вимоги часто змінюються. Ефективна комунікація забезпечує те, що учасникам негайно повідомляють про будь-які зміни, які можуть вплинути на оцінку проекту. Цей проактивний підхід дозволяє учасникам адаптуватися до змінюючихся обставин, сприяючи гнучкому та реагуючому середовищу проекту.

Комунікація з учасниками вирішальна для ідентифікації та вирішення потенційних ризиків. Зберігаючи відкритий канал комунікації, учасники можуть внести цінні відомості про ризики, які можуть бути невидимими під час процесу оцінювання. Ця співпраця підвищує можливості проектного колективу впроваджувати ефективні стратегії мінімізації ризиків, зменшуючи вплив непередбачених труднощів.

Успішні програмні проекти процвітають завдяки співпраці. Ефективна комунікація створює середовище, де учасники та проектні команди спільно працюють над спільними цілями. Розвиваючи культуру співпраці, учасники стають активними в подорожі проекту, сприяючи загальному успіху розробки програмного забезпечення.

Під час оцінювання проекту програмного забезпечення роль комунікації з учасниками є центральною. Чітка та прозора комунікація будує загальне розуміння, сприяє довірі та активно залучає учасників у процес оцінювання. Визнання важливості комунікації з учасниками дозволяє командам інженерії програмного забезпечення управляти складнощами оцінювання з впевненістю, забезпечуючи відповідність результатів проекту очікуванням учасників в постійно змінному середовищі.

1.3.4 Оточуючі та зовнішні фактори в оцінці

Зовнішні та оточуючі чинники мають значущий вплив, створюючи динамічне середовище, в якому оцінювачі повинні прямувати серед невизначеностей поза межами коду та команд розробників.

Економічне середовище може відкинути велику тінь на оцінювання програмного забезпечення. Економічні коливання, рівні інфляції та вартість валюти безпосередньо впливають на вартість ресурсів, праці та технологій. Оцінювачам слід бути пильними до економічних показників, коригуючи оцінки, щоб врахувати можливі варіації вартості, які можуть виникнути через зміни в економічному кліматі.

Проекти програмного забезпечення не є ізольованими сутностями; вони існують у ширшому контексті ринкових тенденцій та вимог. Оцінювачам слід

враховувати зміну потреб та очікувань кінцевих користувачів, ринкових конкурентів та нових технологій. Передбачення змін на ринку, забезпечує те, що оцінки відповідають стратегічним цілям проектів програмного забезпечення та залишаються актуальними в ринковому середовищі, яке стрімко змінюється.

Нормативне середовище може вносити непередбачені складнощі в проекти програмного забезпечення. Зміни в регулюванні галузі, закони про захист даних чи вимоги щодо відповідності можуть вимагати коригувань графіків проекту та розподілу ресурсів. Оцінювачі повинні тримати руку на пульсі регуляторного розвитку, щоб ефективно інтегрувати ці фактори в процес оцінювання.

Темпи технологічних досягнень вводять подвійний виклик та можливість для оцінювання програмного забезпечення. З одного боку, прийняття передових технологій може прискорити розвиток, але, з іншого боку, залежність від фреймворків, що розвиваються, або інструментів може вносити невизначеності. Оцінювачам потрібно оцінювати вплив технологічних тенденцій на реалізацію проекту та робити обґрунтовані прогнози щодо змін у технічному ландшафті.

Проекти програмного забезпечення часто покладаються на зовнішніх постачальників, послуги сторонніх осіб чи спільні партнерства. Оцінювачам слід враховувати потенційні ризики та залежності, пов'язані з зовнішніми сутностями. Зміни у відносинах з постачальниками, перебої в обслуговуванні або несподівані зміни в зовнішніх залежностях можуть впливати на графіки проекту та потреби в ресурсах, що підкреслює необхідність детальної оцінки зовнішніх ризиків.

Глобалізований характер програмної індустрії означає, що геополітичні події та глобальні кризи можуть відбитися на оцінках проекту. Фактори, такі як політична нестабільність, природні катастрофи чи глобальні ситуації здоров'я, можуть порушити ланцюги постачання, впливати на доступність ресурсів та вносити невизначеності, які оцінювачі повинні враховувати у своїх прогнозах.

Соціальні та культурні чинники формують людські аспекти проектів програмного забезпечення. Динаміка команди, стилі комунікації та культурні відмінності можуть впливати на співпрацю та продуктивність. Оцінювачам потрібно бути настроєними на культурний контекст команд розробників, забезпечуючи, що

оцінки враховують вплив соціальних та культурних впливів на графіки та результати проекту.

Зростаючий акцент на екологічну стійкість вводить новий аспект у оцінку програмного забезпечення. Практики “зелених” обчислень та екологічні регуляції можуть впливати на вибір технологій, експлуатацію центрів обробки даних та загальні підходи до проекту. Оцінювачам слід враховувати екологічний слід програмних проектів та вирівнювати оцінки зі стратегічними цілями сталого розвитку.

З урахуванням вроджених невизначеностей, введених зовнішніми чинниками, стає невід'ємною необхідність ефективного управління ризиками та розроблення резервних планів. Оцінювачам слід проактивно ідентифікувати потенційні ризики, пов'язані з екологічними та зовнішніми чинниками, і розробляти стратегії для їхнього зменшення впливу. Це включає сценарне планування, створення резервних бюджетів та збереження гнучкості в графіках проекту.

The dynamic nature of external factors requires a continuous monitoring and adaptation approach in software estimation. Estimators should regularly reassess external influences, staying informed about changes in economic conditions, market trends, regulations, and technological landscapes. This adaptive approach ensures that estimations remain resilient and responsive to the evolving external environment.

Динамічний характер зовнішніх факторів вимагає постійного моніторингу та адаптаційного підходу в оцінюванні програмного забезпечення. Оцінювачі повинні регулярно переоцінювати зовнішні впливи, будучи інформованими про зміни в економічних умовах, ринкових тенденціях, регулюванні та технологічних середовищах. Цей адаптивний підхід забезпечує, що оцінки залишаються стійкими та реагують на зовнішнє середовище, що змінюється.

1.4 Призначення оцінювання

1.4.1 Справжнє призначення оцінювання

Справжнім призначенням оцінювання є не створення точного передбачення майбутнього, а надання корисного інструменту для прийняття рішень та планування.

Оцінки є необхідними в розробці програмного забезпечення, оскільки вони надають базовий рівень для розуміння масштабу і складності проекту. Ця інформація є критичною для прийняття обґрунтованих рішень щодо можливостей проекту, ризиків та потенційного впливу на бізнес-результати. Однак оцінки за своєю природою є невизначеними, оскільки вони базуються на неповній інформації, припущеннях та судженні людини.

Справжнім призначенням оцінювання є не досягнення ідеальної точності, а надання діапазону ймовірних результатів, які можуть бути використані для керування прийняття рішень та планування. Це означає, що оцінки повинні бути реалістичними, прозорими та базуватися на чіткому розумінні масштабу, вимог і обмежень проекту.

Однією з ключових проблем оцінювання є балансування потреби в точності із реальністю невизначеності. Доцільний підхід до оцінювання включає використання різноманітних технік, таких як зверху-вниз, знизу-вверх та експертна оцінка, і їх поєднання для отримання більш точної та надійної оцінки.

Ще одним важливим аспектом оцінювання є комунікація. Оцінювання — це не лише технічний процес, але й соціальний. Точні оцінки вимагають чіткої комунікації та ефективної співпраці між усіма зацікавленими сторонами, включаючи розробників, керівників проекту, бізнес-аналітиків та клієнтів. Важливо встановити загальне розуміння цілей проекту, ризиків та обмежень та включити зацікавлених сторін у процес оцінювання для забезпечення їх підтримки та зобов'язань.

Крім надання інструменту для прийняття рішень та планування, оцінювання також відіграє критичну роль у керівництві та контролі проекту. Оцінки створюють базовий рівень для вимірювання прогресу та ідентифікації можливих проблем, і

можуть бути використані для відстеження виконання проекту відповідно до його цілей та завдань. Постійне оновлення та уточнення оцінок протягом життєвого циклу проекту дозволяє керівникам проектів забезпечити, що проект йде за графіком і відповідає своїм цілям.

Загалом справжнім призначенням оцінювання в розробці програмного забезпечення є не створення ідеальних передбачень, а надання корисного інструменту для прийняття рішень та планування. За допомогою різноманітних технік оцінювання, чіткої комунікації та ефективної співпраці з зацікавленими сторонами, а також постійного оновлення та уточнення оцінок протягом життєвого циклу проекту, проекти розробки програмного забезпечення можуть бути завершені вчасно, в межах бюджету та вдоволеністю всіх зацікавлених сторін.

1.4.2 Переваги точної оцінки

Точні оцінки надають численні переваги, включаючи краще прийняття рішень, краще управління ризиками та підвищений контроль над проектом.

Однією з основних переваг точних оцінок є краще прийняття рішень. Точні оцінки дозволяють учасникам проекту приймати обґрунтовані рішення щодо розподілу ресурсів, обсягу проекту та термінів виконання. Ця інформація критична для забезпечення вчасного завершення проектів, в межах бюджету та задоволення всіх зацікавлених сторін.

Точні оцінки також сприяють кращому управлінню ризиками. Надаючи чітке розуміння вимог проекту, його обсягу та обмежень, точні оцінки допомагають виявляти потенційні ризики і вразливості. Цю інформацію можна використовувати для розробки стратегій мінімізації ризиків та планів на випадок непередбачених обставин, що зменшує ймовірність затримок чи невдачі проекту.

Крім поліпшення прийняття рішень та управління ризиками, точні оцінки також підвищують контроль над проектом. Забезпечуючи базовий рівень для вимірювання прогресу та ідентифікації потенційних проблем, точні оцінки дозволяють керівникам проектів відстежувати виконання проекту відповідно до його цілей та завдань. Цю інформацію можна використовувати для коригування планів

проекту, перерозподілу ресурсів та прийняття інших ключових рішень для збереження проекту на правильному шляху.

Іншою перевагою точних оцінок є поліпшена комунікація зі зацікавленими сторонами. Точні оцінки надають чітке розуміння вимог проекту, термінів виконання та витрат, що дозволяє зацікавленим сторонам ефективно спілкуватися один з одним. Сюди входять розробники, керівники проекту, бізнес-аналітики та клієнти, всі вони відіграють важливу роль у успіху проекту.

Зрештою, точні оцінки також можуть призвести до поліпшення якості програмного забезпечення. Коли оцінки є точними, розробники можуть виділити достатньо часу та ресурсів для тестування, виправлення помилок та забезпечення якості. Це може допомогти забезпечити, що кінцевий продукт відповідає бажаним стандартам якості і не має помилок чи дефектів.

1.4.3 Управління ризиками

Оцінка, за своєю природою, передбачає створення прогнозів про майбутнє. Однак майбутнє в розробці програмного забезпечення є неодмінно невизначеним. Непередбачувані труднощі, зміни вимог та зовнішні фактори можуть значно вплинути на терміни проекту, витрати та потреби в ресурсах. Визнання і прийняття цієї невизначеності є першим кроком до ефективного управління ризиками в оцінці.

Управління ризиками починається з комплексного визначення потенційних ризиків. Це передбачає спільні зусилля серед учасників проекту, включаючи розробників, керівників проекту та навіть кінцевих користувачів. Ризики можуть охоплювати різні аспекти, такі як технічна складність, обмеження ресурсів, динаміка ринку та зовнішні залежності. Ретельне визначення ризиків лягає в основу міцної стратегії управління ризиками.

Після визначення ризиків їх піддають якісному та кількісному аналізу. Якісний аналіз передбачає оцінку впливу та ймовірності кожного ризику, їх категоризацію за ступенем тяжкості. Кількісний аналіз присвоює числові значення ризикам, що дозволяє отримати більш кількісне розуміння їхнього потенційного

впливу на проект. Разом ці оцінки надають відомості для пріоритизації та керування ресурсами для їхнього пом'якшення.

Ефективне управління ризиками передбачає не лише розуміння ризиків, але й розробку стратегій для пом'якшення їхнього впливу. Стратегії пом'якшення можуть варіюватися від коригування термінів проекту та розподілу ресурсів до розробки планів на випадок непередбачених обставин для можливих труднощів. Спільні сесії мозкового штурму та внесок експертів часто сприяють розробці творчих та ефективних стратегій пом'якшення ризиків.

Управління ризиками повинно бути повністю інтегроване в процес планування проекту. Оцінка не є одноразовою дією, а ітеративним процесом, який еволюціонує з розвитком проекту. Регулярне переглядання та оновлення оцінок ризиків забезпечує, що модель оцінки залишається таким, що реагує на зміну обставин та нові ризики.

Учасники проекту повинні бути проінформовані про визначені ризики, потенційні впливи та стратегії для їхнього пом'якшення. Прозора комунікація сприяє культурі довіри та співпраці, вирівнюючи всі сторони, що беруть участь, до спільного розуміння ризиків проекту.

Управління ризиками не завершується із початковим визначенням і плануванням пом'якшення. Неперервний моніторинг протягом життєвого циклу проекту є невід'ємною частиною. Нові ризики можуть виникнути, і ступінь важкості існуючих ризиків може змінитися. Регулярні оцінки ризиків дозволяють командам динамічно адаптувати свої стратегії, забезпечуючи, що проект лишається на правильному шляху перед викликами еволюції.

Кожен проект надає цінні уроки. Після завершення проекту оцінка повинна включати аналіз того, наскільки добре були визначені, управлялися і пом'якшувалися ризики. Цей постійний процес сприяє організаційному навчанню, покращуючи ефективність стратегій управління ризиками в майбутніх проектах.

1.4.4 Етика в практиці оцінювання

Оцінювання, у своїй основі, це не лише технічне завдання, а й практика, вбудована в етичні міркування. Рішення, які приймаються під час оцінювання, можуть мати далекосяжні наслідки, впливаючи на графіки проекту, розподіл ресурсів та очікування зацікавлених сторін. Етична поведінка служить компасом, що направляє оцінщиків через складне середовище невизначеностей та викликів.

Етичне оцінювання надає велике значення прозорості. Надання зацікавленим сторонам чіткого уявлення про процес оцінювання, припущення, зроблені та вбудовані невизначеності, є фундаментальним. Прозорість сприяє довірї між зацікавленими сторонами, дозволяючи їм приймати обґрунтовані рішення та брати активну участь у проектному шляху.

Справедливість є критичним етичним врахуванням при оцінюванні, особливо стосовно розподілу ресурсів. Оцінникам слід прагнути до справедливого розподілу ресурсів, уникаючи упереджень, які можуть підтримувати певні аспекти проекту перед іншими. Справедливий розподіл ресурсів забезпечує, що всі компоненти проекту отримують належну увагу, сприяючи збалансованому та ефективному процесу розробки.

Етичні оцінники дотримуються принципів відповідальності та чесності. Визнання невизначеностей та можливих ризиків, навіть коли це може бути складним, є важливим. Чесна комунікація про обмеження моделей оцінювання та можливість несподіваних труднощів сприяє створенню середовища цілісності та відповідальності.

Етичні оцінщики утримуються від маніпулювання та упереджень. Це включає представлення оцінок на основі об'єктивного аналізу, а не суб'єктивних уподобань. Маніпулювання оцінками, щоб вони відповідали заздалегідь визначеним очікуванням, піддає сумніву цілісність процесу оцінювання, руйнуючи довіру та достовірність.

Етичне оцінювання враховує цінності та пріоритети зацікавлених сторін. Це передбачає активну взаємодію з зацікавленими сторонами для розуміння їх очікувань, обмежень та стратегічних цілей. Співпраця оцінювання з цінностями

зацікавлених сторін гарантує, що результати відповідають їх загальним цілям, сприяючи спільному та взаємовигідному співробітництву.

Перевантаження може бути потенційною етичною пасткою при оцінюванні. Етичні оцінщики уникають спокуси обіцяти більше, ніж реально можна здійснити. Перевантаження може призвести до невиконання очікувань, напружених відносин та можливих невдач проекту. Етична поведінка включає встановлення реалістичних очікувань та забезпечення того, що зобов'язання відповідають фактичним можливостям колективу розробки.

Етичний підхід до оцінювання передбачає зобов'язання до постійного навчання та вдосконалення. Етичні оцінщики активно шукають зворотний зв'язок, вчаться на основі минулих досвідів та вдосконалюють свої методології. Прийняття менталітету росту сприяє постійному розвитку етичних практик оцінювання, адаптуючись до змінного ландшафту розробки програмного забезпечення.

1.5 Вплив на оцінку

1.5.1 Вплив технологій, що розвиваються, на оцінку

У середовищі інженерії програмного забезпечення, що стрімко змінюється, впровадження новітніх технологій перевертає традиційні парадигми, і ніде це перетворення не таке впливове та важливе, як у сфері оцінки.

Штучний інтелект оголошує нову еру у сфері оцінки програмного забезпечення. Алгоритми штучного інтелекту можуть аналізувати великі набори даних, історичну інформацію про проекти та зовнішні фактори для генерації більш точних передбачень. Моделі машинного навчання адаптуються та удосконалюються з часом, постійно вдосконалюючи процес оцінки на основі реальних даних.

Інтеграція аналізу прогнозування в інструменти оцінки дозволяє отримати більш точне розуміння складнощі проекту. За допомогою використання високорозвинених статистичних методів інженери програмного забезпечення можуть передбачити можливі труднощі, оцінити ризики та налаштовувати оцінки з рівнем деталізації, який раніше був недосяжним.

Автоматизація оптимізує робочі процеси оцінки, зменшуючи ручні зусилля і підвищуючи ефективність. Завдання, такі як збір даних, аналіз і генерація звітів, можуть бути автоматизовані, дозволяючи професіоналам з оцінки фокусуватися на більш стратегічних аспектах процесу. Це не лише прискорює час на оцінку, але й мінімізує ризик людських помилок.

Адаптивність машинного навчання особливо впливова в постійно змінному середовищі розробки програмного забезпечення. Алгоритми машинного навчання можуть вчитися на основі відхилень проекту, виявляти патерни і коригувати моделі оцінок в реальному часі. Ця адаптивність підвищує стійкість практик оцінки в динамічних і гнучких проектних середовищах.

Впровадження в оціночні інструменти аспекту визнання патернів, що є основою штучного інтелекту та машинного навчання, відіграє ключову роль у підвищенні точності оцінок. Шляхом визначення патернів у історичних даних проекту інструменти оцінки можуть ідентифікувати схожість, взаємозв'язки та можливі труднощі. Це призводить до більш обґрунтованих прогнозів і глибшого розуміння факторів, які впливають на реалізацію проекту.

Новітні технології сприяють безперервній інтеграції між інструментами оцінки та платформами управління проектами. Ця взаємодія дозволяє приймати більш всебічний підхід до планування проекту, де дані оцінок безпосередньо впливають на розподіл завдань, управління ресурсами та відстеження віхов, сприяючи єдності та даними-орієнтованому середовищу проекту.

Хоча вплив новітніх технологій трансформаційний, він також вносить етичні погляди на перший план. Потенційна упередженість в алгоритмах штучного інтелекту та етичні аспекти використання виключно автоматизованих процесів вимагають ретельного розгляду. Встановлення рівноваги між технологічними досягненнями та етичною відповідальністю є ключовим для стійкої та справедливої практики оцінки.

Майбутнє оцінки програмного забезпечення полягає в співпраці між людською експертизою та можливостями штучного інтелекту. Інтуїція людини, контекстне розуміння та творчість доповнюють аналітичні можливості штучного

інтелекту. Синергія між людським та машинним інтелектом має ключ до розкриття оптимальних результатів в оцінці, де кожен вносить свої унікальні сильні сторони.

Однією з основних рис новітніх технологій у сфері оцінки є їх здатність сприяти безперервному навчанню. Моделі штучного інтелекту та машинного навчання можуть адаптуватися та розвиватися, вивчаючи на кожній ітерації проекту. Цей динамічний процес навчання сприяє постійному вдосконаленню точності та ефективності оцінок з плином часу.

Дивлячись в майбутнє, вплив новітніх технологій на оцінку програмного забезпечення продовжить переосмислення методологій. Злиття штучного інтелекту, машинного навчання та автоматизації, ймовірно, призведе до новаторських підходів, інструментів та фреймворків, формуючи майбутнє, де оцінка стане більш точною, адаптивною та безперешкодно впровадженою в середовище розробки програмного забезпечення.

1.5.2 Культурний та організаційний вплив на оцінку

Вплив організаційного керівництва на практики оцінки не може бути недооціненим. Керівництво встановлює тон для всього процесу розробки програмного забезпечення, впливаючи на пріоритети, цінності та критерії прийняття рішень, які визначають оцінку. Культура прозорості, відповідальності та відповідність стратегічним цілям організації, якій сприяє ефективне керівництво, підвищує надійність результатів оцінки.

Норми комунікації всередині організації відіграють ключову роль у процесі оцінки. Прозорі канали комунікації та відкриті діалоги між членами команди, зацікавленими особами та керівництвом сприяють спільному розумінню складнощів проекту. Прозора комунікація сприяє довірі та дозволяє здійснювати більш точні оцінки, забезпечуючи доступність усієї відповідної інформації для учасників процесу.

Оцінка — це спільне зусилля, і організаційні структури, що сприяють командній роботі та співпраці, позитивно впливають на практики оцінки. Крос-функціональна співпраця, коли залучається внесок з різних відділів та

областей експертизи, надає більш глибоку перспективу вимог проекту. Цей колаборативний підхід зменшує ризик пропусків і забезпечує, що різноманітні точки зору враховуються в процесі оцінки.

Спрямованість культури організації на прийняття ризиків глибоко впливає на практики оцінки. Культури, які сприяють інноваціям і розглядають невдачі як можливості для навчання, створюють середовище, де оцінювачі відчувають себе уповноваженими розглядати більший спектр можливостей. Навпаки, культури, що бояться ризику, можуть призвести до консервативних оцінок, що потенційно ускладнює дослідження більш ефективних чи новаторських підходів.

У середовищі гнучкої розробки, де адаптивність є наріжним каменем, організаційна гнучкість безпосередньо впливає на практики оцінки. Гнучкі методології підкреслюють реагування на зміни над виконання жорсткого плану. Організації, які сприймають цю гнучкість, заохочують оцінювачів коригувати свої прогнози у відповідь на вимоги проекту, що змінюються, сприяючи більш оперативному та точному процесові оцінки.

Оцінка повинна тісно взаємодіяти з організаційними та проектними цілями. Коли культурні та організаційні цінності відповідають цілям конкретного проекту, оцінювачі можуть робити передбачення, які не лише технічно обґрунтовані, але й стратегічно. Ця відповідність гарантує, що оцінки є цінними інструментами для прийняття рішень та розподілу ресурсів відповідно до ширших організаційних цілей.

У світі глобалізації міжкультурні аспекти додають додатковий рівень складності до оцінки програмного забезпечення. Організації з різноманітними командами повинні керувати культурними відмінностями, які можуть впливати на стилі комунікації, норми роботи та очікування. Усвідомлення цих міжкультурних динамік важливе для створення процесів оцінки, які враховують різні точки зору та підходи.

Організаційні структури, зокрема ієрархічні, можуть впливати на потік інформації та прийняття рішень при оцінці. У більш ієрархічних середовищах отримання точних даних та інсайтів від різних членів команди може вимагати

обережного взаємодії з організаційними ієрархіями. Розуміння та пристосування до цих структур є важливим для ефективної оцінки.

Організації, які приймають культуру постійного вдосконалення, сприяють еволюції практик оцінки. Організації, що підтримують вивчення, спонукають до рефлексії над минулими проектами, обміну інсайтами та впровадження вивчених уроків. Цей постійний процес навчання підвищує адаптивність та точність моделей оцінки з плином часу.

1.5.3 Міждисциплінарні погляди на оцінку

Міждисциплінарні погляди, отримані з різних галузей, таких як економіка, психологія та управління проектами, звертають увагу на нові перспективи та підходи до оцінки програмного забезпечення.

Економічні принципи відіграють ключову роль у формуванні оцінок програмного забезпечення. Оцінювачі часто використовують економічні концепції для оцінки часу, ресурсів та вартості можливості, пов'язаних з проектами програмного забезпечення. Принципи попиту та пропозиції, аналіз вартості та динаміка ринку надають основу для кількісної оцінки економічних наслідків різних рішень проекту, впливаючи на результати оцінок.

Область психології розкриває людські аспекти оцінки. Когнітивні упередження, властиві людському прийняттю рішень, можуть значно впливати на точність оцінок. Розуміння упереджень, таких як упередження оптимізму чи якір, дозволяє оцінювачам пом'якшити їх вплив, сприяючи більш об'єктивній та реалістичній оцінці часових рамок та потреб у ресурсах проекту.

Практики управління проектами надають цінні висновки щодо вирівнювання оцінок із методологіями. Незалежно від того, чи слідують традиційним методам каскадного розвитку чи гнучким методологіям Agile, міждисциплінарна інтеграція з принципами управління проектами забезпечує вирівнювання оцінок із загальними цілями проекту. Концепції, такі як WBS та CPS, сприяють загальному розумінню динаміки проекту.

Галузь статистики надає оцінювачам потужні інструменти для аналізу даних. Використовуючи статистичні техніки, оцінювачі можуть визначити закономірності, кореляції та тенденції в історичних даних проекту. Регресійний аналіз, аналіз варіацій та розподіл ймовірностей пропонують кількісні методи для обґрунтованих передбачень, сприяючи точності моделей оцінок.

Принципи оптимізації розподілу ресурсів управлінського обліку сприяють оптимізації розподілу ресурсів у оцінці програмного забезпечення. Оцінювачі можуть використовувати техніки, такі як лінійне програмування чи теорія черг, для моделювання та аналізу розподілу ресурсів, забезпечуючи вирівнювання оцінок із ефективним та стратегічним використанням персоналу, часу та технологій.

Ефективна комунікація — це основа успішних проектів програмного забезпечення, а погляди наукового спілкування збагачують практику оцінки. Оцінювачі можуть скористатися розумінням теорій спілкування, динаміки мови та стратегій залучення стейкхолдерів. Чітка та прозора комунікація сприяє співпраці, вирівнюючи всі сторони з процесом оцінки.

Теорія ігор надає можливість оцінювачам моделювати взаємодії та залежності в проектах програмного забезпечення. Оцінка передбачає участь численних стейкхолдерів із різними інтересами та цілями. Принципи теорії ігор допомагають аналізувати стратегічні взаємодії, потенційні конфлікти та кооперативну динаміку, покращуючи здатність оцінювачів орієнтуватися в складних екосистемах проектів.

Оцінка — це не лише технічна вправа, але і відображення динаміки команди. Уявлення з організаційної психології допомагає розуміти командну співпрацю, мотивацію та патерни комунікації. Оцінювачі можуть застосовувати ці знання для навігації людськими аспектами оцінки, сприяючи створенню середовища, де команди можуть спільно сприяти точним передбаченням.

Правові та етичні врахування є важливими в оцінці програмного забезпечення. Знання правових і етичних аспектів допомагає оцінювачам орієнтуватися в контрактних зобов'язаннях, питаннях інтелектуальної власності та етичних відповідальностях. Розуміння правового положення забезпечує

вирівнювання оцінок із вимогами регулятивних актів та етичними стандартами, пом'якшуючи можливі ризики.

Зростаюча увага до екологічної стійкості вносить погляди з екологічних наук у оцінку програмного забезпечення. Оцінювачі можуть враховувати екологічний вплив проектів програмного забезпечення, вирівнюючи оцінки із цілями сталого розвитку. Цей міждисциплінарний підхід сприяє голістичному підходу, який враховує технічні та екологічні аспекти.

1.6 Метрики - як постійне вдосконалення оцінки

1.6.1 Метрики та вимірювання оцінки

Метрики надають кількісну основу, на якій ґрунтуються практики оцінки. Чи то вимірюючи рядки коду, функціональні бали чи інші відповідні параметри, метрики служать сирими даними, які оцінювачі використовують для аналізу, порівняння та передбачення зусиль, часу та ресурсів, необхідних для проектів з розробки програмного забезпечення. Встановлення міцного фундаменту метрик є важливим для створення оцінок, що ґрунтуються на об'єктивних даних.

Різні метрики відіграють ключову роль у процесі оцінки. Рядки коду, традиційна метрика, кількісно характеризують розмір і складність програмного забезпечення. Функціональні бали пропонують більш абстрактний вимір, зосереджений на функціональності, наданій кінцевим користувачам. Цикломатична складність оцінює заплутаність потоку управління в коді. Кожна метрика надає унікальну перспективу, через яку оцінювачі можуть оцінювати різні аспекти проекту, сприяючи більш комплексному підходу до оцінки.

Історичні дані служать цінним скарбом для оцінювачів. Аналізуючи минулі проекти зі схожими характеристиками, оцінювачі можуть виявляти патерни, тенденції та вказівники, що інформують поточні оцінки. Залежність від історичних даних дозволяє використовувати більш обґрунтований та заснований на даних підхід, зменшуючи залежність від інтуїції та суб'єктивних суджень.

У середовищі гнучкої розробки Agile традиційні метрики можуть потребувати адаптації для вирівнювання з ітеративним і приростовим характером методологій Agile. Метрики Agile, такі як швидкість та графіки виконання (burnout charts), надають відомості про прогрес команди та допомагають оцінювачам динамічно коригувати передбачення. Ці метрики сприяють більш адаптивному та реактивному процесу оцінки в динамічному контексті проектів Agile.

Метрики утворюють будівельні блоки моделей оцінок, які є математичними представленнями відносин між різними параметрами проекту. Моделі оцінок привзодять до системного аналізу та передбаченню, поліпшуючи точність оцінок з плином часу. Визначення та вдосконалення моделей оцінок — це неперервний процес, який користується постійним зворотним зв'язком та коригуванням.

Якість метрик безпосередньо впливає на точність, чіткість та надійність оцінок. Точні метрики відображають справжній характер проекту, чіткість забезпечує послідовність та повторюваність, а надійність надає впевненість в результатах оцінок. Оцінювачі повинні віддавати перевагу вибору та використанню метрик, які відповідають цим критеріям, щоб створювати оцінки, які витримують критику.

Шлях розробки програмного забезпечення динамічний, і метрики надають засіб для постійного моніторингу та зворотного зв'язку. Регулярна оцінка проектних метрик дозволяє оцінювачам виявляти відхилення, відстежувати прогрес та адаптувати оцінки відповідно. Зворотні зв'язки гарантують, що оцінки залишаються відповідними змінам характеру проектів, сприяючи більш реактивному та гнучкому підходу.

Метрики відіграють ключову роль у керуванні ризиками під час оцінки. Визначаючи та кількісно оцінюючи потенційні ризики через метрики, оцінювачі можуть розробляти стратегії превентивного управління ризиками. Метрики, такі як складність коду чи швидкість команди, служать попереджувальними індикаторами, що дозволяють командам вирішувати проблеми, перш ніж вони загострюються, тим самим сприяючи більш стійким та свідомим оціночним практикам.

Метрики виступають потужним інструментом для комунікації результатів оцінки стейкхолдерам. Прозоре надання відповідних метрик, припущень та обґрунтування за оцінками сприяє спільній та свідомий процес прийняття рішень. Ефективна комунікація на основі метрик будує довіру та вирівнює інтереси між командами проекту та стейкхолдерами.

Після завершення проекту метрики відіграють ключову роль у післяпроектному оцінюванні. Порівняння оцінених метрик з фактичними результатами проекту сприяє ретроспективному аналізу, надаючи цінні висновки для постійного навчання та вдосконалення. Цей рефлексивний процес сприяє вдосконаленню моделей та методологій оцінки для майбутніх проектів.

1.6.2 Постійне вдосконалення оцінки

У основі успішної оцінки програмного забезпечення знаходиться культура, яка цінує та приймає постійне вдосконалення. Організації, які надають перевагу вивченню з досвіду, як позитивного, так і негативного, створюють середовище, де оцінювачі мають змогу вдосконалювати методології, вдосконалювати процеси та підвищувати якість практик оцінки.

Одним з ключових принципів постійного вдосконалення в оцінці є ретроспективний аналіз минулих проектів. Вивчення історичних даних, розуміння того, що працювало добре, виявлення областей для вдосконалення та рефлексія щодо розходжень між оцінками та фактичними результатами є важливими кроками в уточненні підходів до оцінки. Цей ретроспективний процес надає непоцінні уроки для майбутніх оцінок.

Постійне вдосконалення передбачає ітеративне уточнення моделей оцінки. Оскільки оцінювачі збирають більше даних та інсайтів, вони можуть вдосконалювати та адаптувати свої моделі, щоб краще відображати тонкощі різних контекстів проектів. Це постійне вдосконалення забезпечує, що моделі оцінок залишаються реактивними на змінну динаміку розробки програмного забезпечення.

Створення зворотних зв'язків є фундаментальним елементом постійного вдосконалення. Ці цикли створюють механізми для коригування оцінок у реальному часі протягом проекту. Активний пошук та врахування зворотного зв'язку від команд

проекту, зацікавлених сторін та кінцевих користувачів дозволяє оцінювачам вносити динамічні коригування, забезпечуючи вирівнювання оцінок із розвитком проекту.

Принципи Agile відіграють важливу роль у постійному вдосконаленні в оцінці. Методології Agile підкреслюють адаптивність, співпрацю та регулярну переоцінку. Оцінювачі, які впроваджують принципи Agile в свої практики, можуть сприяти більш реактивному підходу, коригуючи оцінки на основі ітеративного зворотного зв'язку та приймаючи зміни в вимогах проекту.

Уроки, вивчені з кожного проекту, стають будівельними блоками вдосконалення. Документування та впровадження цих уроків у майбутні практики оцінки запобігає повторенню минулих помилок та сприяє прийняттю успішних стратегій. Цей ітеративний процес вивчення сприяє накопиченню знань, яке зміцнює можливості оцінки.

Постійне вдосконалення передбачає порівняння з найкращими практиками галузі. Оцінювачі можуть порівнювати свої підходи з встановленими стандартами, використовуючи перевірені методології та стратегії. Порівняльний аналіз надає точку відсилання для розуміння, де можна вдосконалитися, сприяючи менталітету відмінності та вирівнюванню із галузевими стандартами.

Технологічні досягнення надають інструменти для автоматизації та отримання інсайтів, забезпечуючи постійне вдосконалення практик оцінки. Інтеграція технологій у процес оцінки дозволяє ефективніше збирати дані, проводити аналіз та уточнення моделей. Автоматизація зменшує ручні зусилля, мінімізує помилки та прискорює темпи вдосконалення.

Постійне вдосконалення розповсюджується на розвиток навичок оцінювання серед членів команди. Програми навчання, семінари та ініціативи з розвитку навичок забезпечують, що оцінювачі будуть в курсі трендів галузі, нових методологій та досягнень в техніках оцінювання. Постійне вдосконалення навичок сприяє гнучкості та пристосованості практик оцінки.

Створення культури зворотного зв'язку в межах процесу оцінки є фундаментом постійного вдосконалення. Сприяючи відкритому спілкуванню, збиранню відгуків від всіх зацікавлених сторін та створенню культури, де зворотний

зв'язок цінується, сприяє динамічному та реактивному середовищу оцінки. Культура зворотного зв'язку дозволяє ідентифікувати можливості для вдосконалення на кожному етапі життєвого циклу оцінки.

1.7 Висновки до розділу

В даному розділі було проведено аналіз підходів до оцінки та прогнозування задач в програмній інженерії. Проведено історичний аналіз оцінки програмного забезпечення та підкреслено важливість графіків розробки програмного забезпечення.

Проведений огляд виявив складнощі, які можуть виникнути при оцінці виконання завдань розробки. Основна увага приділена призначенню оцінювання та впливу на цей процес різноманітних факторів.

У розділі детально розглянуті метрики та засоби постійного вдосконалення вимірювання оцінки, в результаті чого вдалося удосконалити системи оцінювання в розробці програмного забезпечення. Аналіз, проведений у розділі, сприяв покращенню підходів до розв'язання завдань оцінки та прогнозування виконуваності задач у сфері програмної інженерії.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ІСНУЮЧИХ МОДЕЛЕЙ, МЕТОДІВ ТА АЛГОРИТМІВ ПРОГНОЗУВАННЯ ВИКОНУВАНОСТІ ЗАДАЧ В ПРОГРАМНІЙ ІНЖЕНЕРІЇ

2.1 Існуючі моделі оцінювання

2.1.1 Традиційні моделі оцінювання

В галузі інженерії програмного забезпечення давно використовуються традиційні моделі оцінювання для оцінки зусиль, часу та ресурсів, необхідних для завершення проекту. Ці моделі ґрунтуються на встановлених принципах і припущеннях і широко використовуються в різних методологіях розробки програмного забезпечення, таких як модель Waterfall. У цьому розділі ми розглянемо деталі традиційних моделей оцінювання, вивчимо їх базові принципи і оцінимо їхні сильні та слабкі сторони.

Модель Waterfall

Модель Waterfall – це послідовний підхід до розробки програмного забезпечення, який включає в себе визначені етапи, такі як збір вимог, проектування, впровадження, тестування та розгортання [38]. Вперше представлена доктором Вінстоном В. Ройсом в 1970 році, модель Waterfall відображає лінійний і послідовний підхід до розробки програмного забезпечення.

У основі моделі Waterfall лежить прогрес, що йде крок за кроком через визначені етапи. Модель дотримується лінійного і послідовного ходу подій, при цьому кожен етап базується на результаті попереднього. Типові етапи моделі Waterfall включають:

1. Етап вимог: Чітке визначення та документування вимог проекту.
2. Етап проектування: Створення схеми для архітектури програмного забезпечення на основі визначених вимог.
3. Етап впровадження: Перетворення дизайну в конкретний код та розробка програмного продукту.

4. Етап тестування: Поглиблене тестування для виявлення та виправлення дефектів та забезпечення функціональності.

5. Етап розгортання: Випуск програмного продукту для використання кінцевими користувачами.

6. Етап підтримки: Вирішення проблем, внесення поліпшень та забезпечення безперервної функціональності системи.

Модель Waterfall має кілька переваг, що сприяли її тривалій популярності в певних контекстах:

1. Чітка структура проекту: Послідовна природа моделі забезпечує чітку структуру, полегшуючи планування та управління проектом.

2. Документовані процеси: Кожен етап має вичерпну документацію, що полегшує створення комплексної документації проекту.

3. Залучення клієнта: Залучення клієнта зазвичай концентрується на ранніх етапах, забезпечуючи відповідність початковим вимогам.

4. Підходить для стабільних вимог: Коли вимоги проекту стабільні та добре зрозумілі, модель Waterfall може бути дуже ефективною.

Незважаючи на її переваги, модель Waterfall не є позбавленою викликів і критики:

1. Жорсткість: Лінійний характер моделі може бути жорстким, ускладнюючи адаптацію до змін у вимогах проекту.

2. Пізнє виявлення результатів: Клієнти можуть бачити матеріальні результати лише на пізніших етапах, що може призвести до непорозумінь.

3. Обмежена ітерація: Мінімальна кількість ітерацій може викликати труднощі врахування зворотного зв'язку та змін у потребах проекту.

Придатність моделі Waterfall часто залежить від характеру проекту та його вимог. Ця модель особливо підходить для проектів з:

1. Чітко визначеними вимогами: Коли вимоги проекту стабільні та чітко визначені з самого початку.

2. Низькою складністю: Для проектів із низькою складністю та мінімальною невизначеністю.

3. Суворими вимогами до регулювання: В середовищах, де дотримання нормативів є найважливішим.

Згодом практики адаптували модель Waterfall для врахування деяких її обмежень. Варіанти включають:

- Модель V: Варіант, який наголошує на тестуванні на кожному етапі, відображаючи фази розробки.
- Інкрементальна модель: Розробка відбувається порціями, дозволяючи більше гнучкості та зворотного зв'язку.
- Спіральна модель: Включає ітераційні цикли для врахування змінних вимог проекту.

Модель Waterfall має своїх критиків, які вважають, що її негнучкість може бути завадою в динамічному світі розробки програмного забезпечення. Однак прихильники відзначають її придатність для певних контекстів та високо цінують її структурований підхід, підкреслюючи важливість вибору вірної методології на основі вимог проекту.

Переваги традиційних моделей оцінювання

Традиційні моделі оцінки мають кілька переваг у галузі розробки програмного забезпечення [38]. По-перше, вони забезпечують структурований і систематичний підхід до оцінки, забезпечуючи врахування всіх активностей проекту. Лінійна природа моделей дозволяє проводити комплексний аналіз вимог і сприяє деталізованій оцінці на кожному етапі проекту.

По-друге, традиційні моделі оцінки надають відчуття передбачуваності і стабільності. Спираючись на історичні дані та стандарти галузі, ці моделі пропонують точку відліку для оцінки майбутніх проектів. Це може бути особливо корисним для організацій з встановленими процесами та великою кількістю історичних даних проектів.

Крім того, традиційні моделі оцінки часто є більш зрозумілими і доступними, що робить їх доступними як для досвідчених оцінювачів, так і для учасників

проекту. Простота цих моделей дозволяє чіткіше спілкуватись та сприяє прийняттю рішень на основі оцінених термінів та потреб у ресурсах.

Обмеження традиційних моделей оцінювання

Незважаючи на їх широке використання, традиційні моделі оцінки мають суттєві обмеження. Одним із ключових викликів є їх залежність від детального планування та документації на початковому етапі, що передбачає, що всі вимоги проекту повністю відомі та зрозумілі відразу. На практиці вимоги часто еволюціонують протягом життєвого циклу розробки програмного забезпечення, що робить початкові оцінки менш точними по мірі розвитку проекту.

Традиційні моделі оцінки також схильні недооцінювати вплив невизначеності та ризику. Вони можуть недостатньо враховувати зовнішні фактори, такі як зміни ринкових умов або технологічні досягнення, які можуть внести несподівані складнощі та вплинути на терміни та потреби у ресурсах проекту.

Крім того, традиційні моделі оцінки часто мають труднощі із врахуванням варіацій у складі команди, рівнях досвіду та індивідуальній продуктивності. Вони передбачають рівномірний розподіл зусиль і продуктивності серед членів команди, ігноруючи вплив індивідуальних здібностей та навичок на результати проекту.

Інше обмеження полягає в лінійній природі традиційних моделей оцінки, яка може не добре взаємодіяти з гнучкими методологіями розробки або проектами, які вимагають гнучкості та ітеративних зворотних зв'язків. Жорстка структура цих моделей може ускладнити адаптацію до змінних вимог та призвести до неточних оцінок для гнучких або динамічних проектів.

2.1.2 Техніки оцінювання Agile

Гнучкі методології розробки програмного забезпечення набули значного популярності в останні роки завдяки своїй гнучкості, адаптивності та ітеративному характеру [39]. Техніки Agile оцінювання виникли як альтернатива традиційним моделям оцінки, вирішуючи обмеження жорсткого та передбачуваного планування. У цьому розділі ми розглянемо різні техніки Agile оцінювання, які

використовуються в інженерії програмного забезпечення та проаналізуємо їх ефективність у покращенні точності оцінювання.

Оцінка User Story

Оцінка User Story - це широко використовувана техніка в гнучкій розробці, особливо в таких фреймворках, як Scrum [40]. У цьому підході вимоги проекту фіксуються у вигляді користувацьких історій, які представляють конкретні функції або можливості, бажані кінцевими користувачами. Оцінка проводиться шляхом призначення балів історії кожній користувацькій історії, що вказує на її відносний розмір і складність.

Оцінка користувацьких історій не стосується точних вимірювань часу, але є відносною оцінкою зусиль та складності. Цей процес дозволяє гнучким командам оцінювати зусилля, необхідні для кожної користувацької історії, сприяючи спільному розумінню серед членів команди та зацікавлених сторін.

Гнучкі команди часто використовують техніки відносного розміщення для оцінки користувацьких історій [41]. Серед них:

1. Покер планування: Члени команди призначають бали користувацьким історіям за допомогою колоди карт, сприяючи обговоренню та узгодженню.

2. Послідовність Фібоначчі: Призначення балів за допомогою послідовності Фібоначчі (1, 2, 3, 5, 8, 13 та ін.), щоб відобразити зростаючий рівень складності.

3. Розміри футболок: Опис користувацьких історій за допомогою розмірів футболок (маленькі, середні, великі), щоб вказати відносний обсяг зусиль.

Оцінка користувацьких історій прагне до співпраці. Гнучкі команди, які складаються з розробників, тестувальників, власників продукту та інших зацікавлених сторін, об'єднуються для колективної оцінки зусиль, необхідних для кожної користувацької історії. Цей спільний підхід сприяє спільній відповідальності та більш точному процесу оцінювання.

Незважаючи на свої переваги, оцінка користувацьких історій стикається з викликами, включаючи:

1. Різні інтерпретації: Члени команди можуть інтерпретувати користувацькі історії по-різному, що може призводити до розбіжностей в оцінках.

2. Змінні вимоги: Динамічні вимоги проекту можуть впливати на точність оцінок користувацьких історій.

3. Часові обмеження: Обмежений час для сесій оцінювання може вплинути на ретельність процесу.

Оцінка користувацьких історій має численні переваги, серед яких:

1. Покращене планування: Точна оцінка поліпшує планування проекту, дозволяючи командам ефективно розподіляти ресурси.

2. Прозорість: Зацікавлені сторони отримують видимість зусиль, необхідних для кожної користувацької історії, сприяючи прозорості та обгрунтованому прийняттю рішень.

3. Пріоритизація: Оцінка допомагає визначити пріоритети користувацьких історій за їх складністю та цінністю, дозволяючи командам фокусуватися на важливих завданнях першими.

Методології Agile підкреслюють адаптивність, і оцінка користувацьких історій ідеально вписується у цей принцип. Під час еволюції проектів команди переглядають та коригують свої оцінки, адаптуючись до змін вимог та зберігаючи гнучкість перед зміною пріоритетів.

Важливим показником в гнучких методологіях є швидкість (velocity), яка відображає обсяг роботи, який команда може виконати протягом конкретної ітерації чи спринта. Оцінка користувацьких історій вносить вклад у розрахунок швидкості, дозволяючи командам планувати та прогнозувати майбутні спринти на основі історичної продуктивності.

Оцінка користувацьких історій - це не статичний процес, а постійний шлях до вдосконалення. Гнучкі команди регулярно проводять ретроспективи для визначення точності своїх оцінок, виявлення областей для вдосконалення та удосконалення своїх технік оцінювання з часом.

При настанні епохи Agile з'явилися інструменти та платформи, які оптимізують процес оцінювання користувацьких історій. Гнучкі команди

використовують спеціальні інструменти, що сприяють віртуальним сесіям оцінювання, поліпшують співпрацю та зберігають цифровий запис оцінок для майбутнього використання.

Відносне вимірювання (Relative Sizing)

Відносне вимірювання - це ще одна техніка гнучкої оцінки, яка акцентує на порівнянні зусиль, необхідних для різних завдань проекту або користувацьких історій, замість надання абсолютних оцінок [41]. Члени команди призначають розмір або вагу кожному завданню або історії на основі їх сприйнятої складності чи зусиль в порівнянні з базовим завданням або історією. Ця техніка надає відносне розуміння зусиль і допомагає визначати пріоритети та планувати проектні заходи.

Популярний метод відносного розміщення - "Planning Game", де члени команди спільно впорядковують завдання чи користувацькі історії за шкалою складності чи зусиль [40]. Ця техніка сприяє залученню команди, сприяє обміну знанням і дозволяє ефективно визначати пріоритети у беклогу.

Покер Планування (Planning Poker)

Planning Poker - це колективна техніка оцінки, яка передбачає, що члени команди надають свої індивідуальні оцінки для користувацьких історій або завдань проекту [41]. Кожен член команди утримує набір гральних карт, які зазвичай представляють різні числа Фібоначчі або значення балів історії. Модератор оголошує користувацьку історію, і члени команди одночасно показують свої карти, вказуючи свої оцінки зусиль.

Якщо є широкий розрив в оцінках, члени команди обговорюють свої обгрунтування та інсайти, щоб досягти консенсусу. Процес продовжується до тих пір, поки команда не збере конвергенцію за оцінкою зусиль. Planning Poker сприяє прозорості, колективному прийняттю рішень та спільному розумінню серед членів команди.

Wideband Delphi

Wideband Delphi надихається стародавньою грецькою провидцем, перетворюючи непередбачуване в структурований та колективний прогноз. Це є структурованою технікою оцінювання на основі консенсусу, яка використовує колективний інтелект різноманітної команди для формування обґрунтованих та реалістичних оцінок.

У центрі Wideband Delphi - це співпраця. На відміну від традиційних методів оцінювання, де один експерт може нести вагу передбачення, Wideband Delphi запрошує на робочий стіл різноманітну групу зацікавлених сторін. Розробники, експерти з предметної області та менеджери проекту беруть участь у діалозі, вносячи свої перспективи та інсайти у процес оцінювання.

Wideband Delphi розгортається в ітеративних раундах оцінювання та обговорення. Процес починається з початкового раунду, де надаються індивідуальні оцінки анонімно. Ця анонімність звільняє учасників від можливих упереджень та сприяє відвертому внеску. Наступні раунди включають групові обговорення, які сприяють модератор, де учасники можуть поділитися своїми раціоналами, інсайтами та врахуваннями.

Wideband Delphi процвітає на тонкому балансі між розходженням та збіжністю. Початкова анонімність дозволяє розходженню думок, охоплюючи різноманітні погляди в команді. Наступні раунди, відзначені відкритими обговореннями, сприяють конвергенції, оскільки учасники вирівнюють свої точки зору, будуючи спільне розуміння завдання.

Кваліфікований модератор - це диригент, який веде оркестр Wideband Delphi. Модератор забезпечує збереження фокусу обговорень, підтримує активну участь та керує командою через ітеративні цикли оцінювання. Ця модерація має ключове значення для створення середовища, де може розквітати колективний інтелект.

Wideband Delphi виходить за межі числових результатів оцінок. Він вирощує спільне розуміння серед членів команди, сприяючи відчуттю власності та співпраці. Прозорість та інклюзивність методу сприяють більшому залученню та мотивуванню команді, вирівнюючи зусилля на спільну мету.

У сфері оцінювання проектів невідомість неухильна. Wideband Delphi, зі своєю колективною сутністю, вирішує ці невизначеності, звертаючись до різноманітної експертизи всередині команди. Він визнає, що різні точки зору збагачують процес оцінювання, надаючи більш глибокий огляд труднощів та можливостей, що inherent у завданні.

Wideband Delphi виходить за межі конкретних галузей. Його адаптивність робить його цінним інструментом в різних сферах, від розробки програмного забезпечення до будівельних проектів. Залежність методу від колективної мудрості та ітеративного вдосконалення відповідає динаміці завдань у різних галузях.

Wideband Delphi є еволюцією від оригінального методу Дельфі, представленого в середині 20-го століття. Тоді як метод Дельфі спирався на серію анкет для збору та узагальнення експертних думок, Wideband Delphi вводить колективний та інтерактивний елемент, використовуючи потужність обговорень у реальному часі та ітеративного вдосконалення.

Незважаючи на його переваги, Wideband Delphi не обійшовся без викликів. Успіх методу великою мірою залежить від якості модерації, відкритості учасників та доступності різноманітних точок зору. Збалансування потреби в консенсусі з можливістю групового мислення вимагає тонкого підходу.

Wideband Delphi, з акцентом на співпрацю, ітеративне вдосконалення та колективний інтелект, виходить як гармонійний підхід до оцінювання проектів. В ситуації, де чекають невизначеності і складнощі, цей метод організовує симфонію різноманітних голосів, направляючи команди до більш обґрунтованих, включних та реалістичних оцінок. З розвитком проектів і розкриттям викликів Wideband Delphi стоїть як маяк, який освітлює шлях до успішних і спільних результатів проектів.

Оцінка Kanban

Канбан, методологія легкої розробки програмного забезпечення, пропонує унікальний підхід до оцінювання [42]. Замість надання передбачуваних оцінок заздалегідь, Канбан акцентує увагу на візуалізації та управлінні потоком роботи.

Кожне завдання або користувацька історія представлена карткою на дошці Канбан, яка проходить через різні етапи робочого процесу.

Центральним елементом Канбану є візуальне відображення роботи на дошці Канбан. Завдання, представлені у вигляді карток, переміщуються через колонки, які позначають різні етапи робочого процесу. Оцінювання в Канбан використовує цю візуальну структуру, дозволяючи командам оцінити стан роботи на льоту та сприяти обґрунтованому прийняттю рішень.

Канбан визнає, що не всі робочі одиниці створюються рівними. Різні класи обслуговування, що представляють різні типи робіт, дозволяють командам пристосовувати свої методи оцінювання. Наприклад, рутинне завдання може мати інший процес оцінювання, ніж складний розвиток функцій.

В області Канбану цикл часу - тривалість завдання від початку до завершення - виступає в центрі уваги. За допомогою постійного вимірювання та аналізу часу циклу команди отримують інформацію про свою історичну продуктивність, що дозволяє зробити більш точні оцінки та визначити напрямки для вдосконалення.

Канбан працює на засадах систем витягування, де робота витягується в систему на підставі фактичного попиту. Оцінювання в Канбан відповідає цій філософії витягування, дозволяючи командам зосередитися на завершенні існуючої роботи перед витягуванням нових завдань. Цей динамічний підхід дозволяє командам швидко адаптуватися до зміни пріоритетів.

В основі Канбану лежить зобов'язання до постійного вдосконалення. Оцінювання в Канбан не є одноразовою подією, але тривалим ітеративним процесом. Команди регулярно перевіряють та адаптують свої практики оцінювання, уточнюючи своє розуміння роботи та покращуючи свою здатність передбачити результати.

Work in Progress (WIP) ліміти, фундаментальний аспект Канбану, обмежує кількість завдань, дозволених на кожному етапі робочого процесу. Обмеження WIP дозволяє командам підтримувати стабільний потік роботи та уникати перевантаження. Ця обмеженість сприяє більш реалістичному оцінюванню, оскільки команди можуть краще оцінити свою спроможність.

Оцінювання в Канбані втілює принцип Agile відповіді на зміни над виконанням плану. Його гнучкий характер дозволяє командам адаптувати свої оцінки в реальному часі, коли з'являється нова інформація або змінюються пріоритети. Ця адаптивність особливо цінна в динамічних та швидкозмінних проектних середовищах.

Оцінювання в Канбані надає великий акцент на доставку вартості клієнту. Співставляючи оцінки з пріоритетами клієнта та постійно переоцінюючи цінність робочих одиниць, команди можуть забезпечити, що їхні зусилля мають значущий внесок у загальні цілі проекту.

Оцінювання в Канбані здійснюється за допомогою аналізу історичного часу циклу або часу виконання схожих завдань або історій [43]. Слідкуючи за часом, який потрібно картці для переходу від одного етапу до іншого, команди отримують відомості про середню тривалість для конкретних типів робіт. Цей підхід до оцінки, заснований на даних, забезпечує більш реалістичне розуміння часу, необхідного для завершення проекту.

2.1.3 Підходи машинного навчання в оцінюванні

Під час того як проекти в галузі інженерії програмного забезпечення стають більш складними та динамічними, традиційним моделям оцінювання може бути важко надавати точні прогнози [44]. Останнім часом в галузі оцінювання програмного забезпечення набирають популярності підходи Machine Learning (ML), використовуючи потужність аналізу даних та розпізнавання шаблонів. У цьому розділі ми розглянемо застосування методів машинного навчання в інженерії програмного забезпечення для оцінювання та вивчимо їхні можливі переваги та виклики.

Оцінювання на основі даних (Data-Driven Estimation)

Методи оцінювання на основі машинного навчання ґрунтуються на історичних даних проектів для створення прогностичних моделей [45]. Аналізуючи історичні дані проектів, включаючи фактори, такі як розмір проекту, складність, склад команди та середовищні фактори, алгоритми машинного навчання можуть

вивчати шаблони та кореляції для створення моделей оцінювання. Ці моделі можуть використовуватися для прогнозування зусиль, вартості чи тривалості майбутніх проектів.

Наявність великих наборів даних та досягнення в алгоритмах машинного навчання, таких як регресія, дерева рішень та нейронні мережі, дозволяють використовувати дані для оцінювання [46]. Моделі машинного навчання можуть враховувати нелінійні взаємозв'язки та виявляти витончені залежності між різними факторами проекту, що призводить до більш точних та персоналізованих оцінок.

Вибір функцій та розробка

У методах оцінювання на основі машинного навчання важливо визначити та оптимізувати вибір релевантних функцій для ефективності моделі [47]. Важливо визначити найвпливовіші фактори, які вносять вклад у результати проекту. Знання галузі та експертне оцінювання є цінними при виборі функцій, які мають значущий вплив на зусилля чи тривалість проекту.

Розробка функцій передбачає перетворення та поєднання сирих даних в значущі функції, які краще відображають внутрішні зв'язки [48]. Наприклад, замість використання сирих рядків коду, похідні метрики, такі як точки функцій або міри складності, можуть бути більш відповідними для цілей оцінювання. Ретельний вибір та інженерія функцій можуть підвищити передбачувальну силу моделей машинного навчання.

Навчання та перевірка моделі

Моделі машинного навчання потребують тренування на історичних даних для вивчення звичаїв та здійснення передбачень [49]. Процес тренування включає розділення набору даних на тренувальний та валідаційний набори. Модель навчається на тренувальному наборі та перевіряється на валідаційному для оцінки її ефективності та загальної здатності узагальнення.

Для забезпечення надійності та точності моделей важливо використовувати надійні техніки валідації, такі як крос-валідація чи валідація з утриманням [50]. Ці техніки допомагають оцінити продуктивність моделі на невидимих даних та

зменшити проблеми, такі як перенавчання, коли модель стає занадто спеціалізованою на тренувальних даних та погано працює на нових даних.

Виклика та міркування

Хоча підходи машинного навчання відкривають перспективи для оцінки в інженерії програмного забезпечення, є кілька викликів, які потрібно врахувати. Один із значущих викликів - це доступність та якість історичних даних. Достатньо і релевантної інформації є вирішальною для тренування точних моделей машинного навчання. У деяких випадках організації можуть потребувати збирати та обробляти додаткові дані для покращення здатностей оцінки.

Також важливим є інтерпретованість та пояснюваність моделей машинного навчання. Моделі машинного навчання, особливо складні, такі як нейронні мережі, можуть вважатися чорними скриньками, що ускладнює розуміння та пояснення факторів, що впливають на оцінку. Це може бути проблемою, особливо в регульованих галузях чи в ситуаціях, де потрібна прозорість.

Ще однією розглядуваною проблемою є необхідність постійного оновлення та адаптації моделей. Проекти з програмного забезпечення змінюються з часом, і моделі оцінки повинні оновлюватися, щоб відображати змінюючуся динаміку. Регулярне перетренування моделей з оновленими даними забезпечує їх актуальність та точність.

2.1.4 Параметрична модель оцінювання

Моделі параметричної оцінки широко використовуються в інженерії програмного забезпечення для оцінки зусиль, вартості та тривалості проекту на основі історичних даних та попередньо визначених параметрів. Ці моделі встановлюють взаємозв'язки між характеристиками проекту та бажаними метриками оцінки, що дозволяє проводити ефективні та стандартизовані процеси оцінювання. У цьому розділі ми розглянемо концепцію моделей параметричної оцінки та дослідимо їх переваги, обмеження та застосування в інженерії програмного забезпечення.

Огляд параметричних моделей оцінювання

Моделі параметричної оцінки використовують математичні формули або статистичні методи для оцінки атрибутів проекту. Ці моделі встановлюють функціональний зв'язок між метрикою оцінки (такою як зусилля чи тривалість) та атрибутами проекту (такими як кількість рядків коду, функціональні точки чи показники складності) [51]. Квантифікуючи вплив різних факторів проекту, параметричні моделі забезпечують систематичний та повторюваний підхід до оцінки.

Моделі на основі рядків коду (Lines of Code-Based Models)

Один із широко використовуваних підходів параметричної оцінки - це моделі, що базуються на кількості Lines of Code (LOC) [51]. Ці моделі оцінюють зусилля та тривалість проекту на основі кількості рядків коду. Аналізуючи історичні дані з аналогічних проектів, застосовують регресійний аналіз чи статистичні методи для похідної формули або рівняння, яке пов'язує LOC зі зусиллям чи тривалістю.

Моделі, що базуються на LOC, пропонують простоту та зручність використання, оскільки підрахунок рядків коду - це прямолінійний процес. Однак вони мають свої обмеження, оскільки лише кількість рядків коду може не враховувати аспекти складності та якості розробки програмного забезпечення. Крім того, моделі на основі LOC передбачають лінійний зв'язок між розміром коду та зусиллям, що не завжди відповідає практиці.

Аналіз функціональних точок (Function Point Analysis)

Function point analysis (FPA) - це широко використовувана техніка для оцінювання розміру та зусиль програмного забезпечення [52]. FPA вимірює функціональність, яку забезпечує програмна система на основі вимог користувача. Функціональні точки враховують такі фактори, як введення, виведення, запити, файли та інтерфейси, щоб призначити числові значення різним компонентам системи.

Параметричні моделі оцінки, засновані на функціональних точках, використовують історичні дані для встановлення взаємозв'язків між

функціональними точками та зусиллям чи тривалістю. Ці моделі надають більш абстрактний та високорівневий підхід до оцінювання, зосереджений на функціональній складності програмної системи.

Моделі на основі програмних метрик (Software Metrics-Based Models)

Параметричні моделі оцінювання також можуть використовувати програмні метрики, такі як показники складності, щільність дефектів або цикломатична складність, для оцінки зусиль чи тривалості проекту [53]. Аналізуючи історичні дані та визначаючи взаємозв'язок між програмними метриками та потрібними метриками оцінювання, ці моделі забезпечують більш деталізований та контекст-специфічний підхід до оцінювання.

Моделі на основі програмних метрик вимагають уважного вибору відповідних метрик, які враховують аспекти складності, збереженості та якості програмного забезпечення. Ці моделі можуть бути адаптовані до конкретних парадигм розробки програмного забезпечення чи областей, що дозволяє здійснювати більш точні оцінки в конкретних контекстах.

Сильні сторони та обмеження

Параметричні моделі оцінювання мають кілька переваг у сфері оцінювання програмного забезпечення. Вони надають структурований та стандартизований підхід, що дозволяє організаціям встановлювати процеси та бенчмарки оцінювання. Ці моделі використовують історичні дані, що дозволяє здійснювати оцінки на основі реальних досвідів з проектів [54]. Параметричні моделі є масштабованими, оскільки їх можна застосовувати до проектів різного розміру та складності.

Однак параметричні моделі мають обмеження, які слід враховувати. Вони передбачають статичний зв'язок між атрибутами проекту та метриками оцінювання, що може не враховувати еволюцію практик розробки програмного забезпечення чи технологічні досягнення. Параметричні моделі також сильно залежать від історичних даних, і їх точність може бути позначено змінами у контексті проекту чи технологіях. Важливо періодично перевіряти та оновлювати параметричні моделі новими даними для збереження їх актуальності та точності.

2.1.5 Статистичні техніки оцінювання

Статистичні техніки оцінювання відіграють важливу роль у програмному інженерії для здійснення точних прогнозів та проекцій на основі аналізу історичних даних. Ці техніки використовують статистичні моделі, алгоритми та принципи для оцінки атрибутів проекту, таких як зусилля, вартість і тривалість. У цьому розділі ми розглянемо різноманітні статистичні техніки оцінювання, що широко використовуються в програмному інженерії, і обговоримо їхні переваги, обмеження та застосування.

Регресивний аналіз (Regression Analysis)

Регресивний аналіз - це широко використовувана статистична техніка для оцінки взаємозв'язків між змінними. У програмному інженерії аналіз регресії використовується для встановлення функціонального відношення між атрибутами проекту (незалежні змінні) та метриками оцінки (залежні змінні) [55]. Аналізуючи історичні дані, моделі регресії можуть передбачати значення метрик оцінки на основі заданих атрибутів проекту.

Проста лінійна регресія - це найбільш базова форма аналізу регресії, що передбачає лінійне відношення між незалежними та залежними змінними. Множинна лінійна регресія розширює цей концепт, включаючи кілька незалежних змінних. Також використовуються нелінійні моделі регресії, коли відношення між змінними є нелінійним.

Аналіз регресії надає кількісний підхід до оцінювання і може бути застосований в різних контекстах програмного інженерії. Однак він передбачає, що відношення між змінними залишається постійним, що може не відповідати складним програмним проектам із змінними динаміками [56].

Аналіз часових рядків (Time Series Analysis)

Аналіз часових рядів зосереджений на вивченні та передбаченні закономірностей у даних протягом часу [57]. Він особливо корисний для оцінки атрибутів проекту, які мають тимчасові залежності, таких як зусилля або кількість

дефектів. Моделі часових рядів захоплюють тенденції, сезонність та інші залежності від часу, щоб здійснювати майбутні прогнози [58].

Загальні моделі часових рядів включають Autoregressive Integrated Moving Average (ARIMA), моделі експоненційного згладжування (наприклад, Holt-Winters) та сезонний розклад часового ряду. Ці моделі можуть захоплювати короткострокові коливання та довгострокові тенденції в даних проекту, що дозволяє здійснювати точні оцінки та прогнози.

Аналіз часових рядів є корисним у випадках, коли доступні історичні дані та коли оцінкова метрика піддається впливу часово залежних факторів. Однак він може не враховувати раптових змін чи неперіодичних подій, які впливають на динаміку проекту, що вимагає додаткових урахувань у таких випадках.

Monte Carlo симуляція

Симуляція Монте-Карло - це ймовірнісний метод, який використовується для моделювання невизначеності та ризику в оцінках програмної інженерії [59]. Він генерує кілька сценаріїв, повторно вибираючи з імовірнісних розподілів проектних змінних та моделюючи результати для оцінки бажаного атрибуту проекту.

З урахуванням варіативності та розподілу проектних змінних симуляція Монте-Карло забезпечує комплексний підхід до оцінки. Вона може враховувати невизначеність, таку як коливання вимог, непередбачувані ризики чи доступність ресурсів, які можуть впливати на результати проекту.

Симуляція Монте-Карло особливо корисна, коли параметри проекту мають значний ступінь невизначеності, а оцінкова метрика впливає на кілька випадкових факторів. Однак для складних проектів вона вимагає глибокого розуміння основних імовірнісних розподілів і може бути витратною з обчислювальної точки зору.

Оцінка Байєса

Байєсівська оцінка - це статистичний підхід, який поєднує попередні знання або переконання зі спостереженнями, щоб робити ймовірнісні оцінки [60]. Вона використовує теорему Байєса для оновлення попередніх переконань на основі нових

доказів, що призводить до апостеріорного розподілу, який представляє оцінку атрибуту проекту.

Байєсівська оцінка надає гнучкий фреймворк для включення попередньої інформації, експертних думок та історичних даних в процес оцінки. Вона дозволяє ітеративне оновлення при наявності нових даних, що забезпечує адаптивну оцінку в динамічних проектах програмного забезпечення.

Однак байєсівська оцінка вимагає точних попередніх знань та відповідного вибору попередніх розподілів, що може бути важким на практиці. Вона також покладається на припущення, що попередні та апостеріорні розподіли слідують певними ймовірнісними розподілами, що не завжди вірно.

2.2 Гібридні підходи та комбіновані моделі

Гібридні підходи та техніки комбінування моделей здобули увагу в області інженерії програмного забезпечення для покращення точності оцінок та прогнозів. Ці підходи інтегрують кілька моделей оцінки, методів або алгоритмів для використання їхніх індивідуальних переваг та компенсації їхніх обмежень. У цьому розділі ми розглянемо гібридні підходи до оцінки та обговоримо, як комбінування моделей може підвищити точність оцінок в інженерії програмного забезпечення.

2.2.1 Гібридні підходи до оцінювання

Гібридні підходи до оцінки спрямовані на поєднання переваг різних моделей оцінки або технік для досягнення більш точних прогнозів [61]. Інтегруючи доповнюючі моделі, ці підходи можуть вирішувати обмеження окремих моделей та охоплювати ширший спектр характеристик проекту.

Наприклад, гібридний підхід може поєднувати параметричну модель із алгоритмом машинного навчання. Параметрична модель може надавати структуровану та засновану на даних оцінку на основі історичних даних, тоді як

алгоритм машинного навчання може виявляти складні взаємозв'язки чи нелінійні залежності, які можуть існувати в даних.

Гібридні підходи також можуть включати експертну оцінку, використовуючи інсайти та досвід фахівців у галузі. Експертна оцінка може бути використана для тонкого налаштування чи коригування оцінок, які генеруються іншими моделями, з урахуванням знань у галузі та контекстуальних факторів, які можуть впливати на результати проекту.

2.2.2 Техніки комбінування моделей

Техніки комбінування моделей передбачають агрегацію прогнозів кількох моделей оцінки для отримання більш точної та стійкої оцінки [62]. Ці техніки спрямовані на зменшення впливу викривлення та дисперсії окремих моделей, використовуючи колективні знання різноманітних моделей.

Ансамбль-методи, такі як bagging (вибірка), boosting (підсилення) чи stacking (укладання), часто використовуються для комбінування моделей. Bagging комбінує прогнози кількох моделей шляхом усереднення чи голосування, тоді як boosting надає ваги моделям на основі їх продуктивності і відповідно комбінує їх прогнози. Stacking передбачає тренування метамоделі, яка вивчає комбінувати прогнози кількох базових моделей.

Техніки комбінування моделей можуть покращити точність оцінки, зменшуючи вплив викидів, пом'якшуючи обмеження окремих моделей та охоплюючи ширший спектр характеристик проекту. Ці техніки особливо корисні, коли окремі моделі мають доповнюючі сильні та слабкі сторони чи коли існує невизначеність у виборі єдиної найкращої моделі.

2.2.3 Міркування та виклики

Хоча гібридні підходи та техніки комбінування моделей можуть мати потенційні переваги, існують кілька обставин та викликів, на які варто звертати увагу:

1. Сумісність моделей: При комбінуванні моделей важливо забезпечити сумісність між ними щодо вхідних змінних, припущень та основних принципів. Несумісні моделі можуть призвести до послідовних або ненадійних оцінок.

2. Вимоги до тренувальних даних: Техніки комбінування моделей зазвичай вимагають достатньої кількості тренувальних даних для навчання та перевірки окремих моделей. Доступність та якість даних є важливими для ефективного комбінування моделей.

3. Складність та інтерпретованість: Гібридні підходи та техніки комбінування моделей можуть внести додаткову складність у процес оцінки. Може бути важко інтерпретувати об'єднані результати чи розуміти внесок кожної окремої моделі.

4. Перенавчання: Слід уникати перенавчання, коли об'єднана модель працює добре на тренувальних даних, але не може узагальнити результати на нові, невидані дані. Техніки регуляризації та перехресної перевірки можуть допомогти зменшити ризики перенавчання.

5. Обслуговування та оновлення: Гібридні підходи та техніки комбінування моделей потребують постійного обслуговування та оновлень. При доступності нових даних або зміні контекстів проекту може знадобитися повторне навчання або налаштування об'єднаних моделей для забезпечення їхньої актуальності [63].

2.3 Порівняльна характеристика моделей оцінювання задач

Оцінка та порівняння моделей оцінювання відіграють важливу роль у оцінці їх ефективності, визначенні сильних та слабких сторін, а також виборі найбільш відповідної моделі для конкретного контексту інженерії програмного забезпечення. У цьому розділі ми розглянемо різні техніки та метрики оцінювання, які використовуються для оцінки ефективності моделей оцінювання і надамо рекомендації для порівняння різних моделей.

2.3.1 Метрики оцінювання

Для оцінки моделей оцінювання зазвичай використовують кілька метрик для вимірювання їх продуктивності. Ці метрики надають інформацію щодо точності, точності та надійності оцінених значень. Деякі загальновживані метрики оцінювання включають:

1. Mean Absolute Error (MAE): MAE вимірює середню абсолютну різницю між оціненими значеннями та фактичними значеннями. Нижчий показник MAE свідчить про кращу точність оцінки.

2. Root Mean Squared Error (RMSE): RMSE обчислює квадратний корінь середньоквадратичних різниць між оціненими значеннями та фактичними значеннями. Він відкидає великі похибки порівняно з MAE [64].

3. Mean Percentage Error (MPE): MPE вимірює середню відсоткову різницю між оціненими значеннями та фактичними значеннями. Він допомагає оцінити відхилення в оцінці.

4. Coefficient of Determination (R^2): R^2 представляє частку варіації залежної змінної, яку можна пояснити незалежними змінними. Вищі значення R^2 свідчать про кращу придатність моделі.

5. Точність та повторюваність: Точність вимірює частку правильно оцінених значень серед передбачених значень, тоді як повторюваність вимірює частку правильно оцінених значень серед фактичних значень. Ці метрики корисні для оцінки моделей оцінювання на основі класифікації.

6. Receiver Operating Characteristic (ROC): Крива ROC відображає співвідношення правильних позитивів і правильних фальшивих позитивів, надаючи інформацію про класифікаційну продуктивність моделі.

2.3.2 Техніки перехресної валідації

Техніки перехресної валідації широко використовуються для оцінки моделей оцінювання, оцінюючи їх продуктивність на невидимих даних. Ці техніки допомагають оцінити, наскільки добре модель узагальнюється до нових, невидимих

екземплярів. Деякі широко використовувані техніки перехресної валідації включають:

1. K-Fold Cross-Validation: Набір даних розділяється на K підмножин, або фолдів. Модель навчається на K-1 фолдах і тестується на залишковому фолді. Цей процес повторюється K разів, при цьому кожен фолд служить тестовим набором даних один раз. Результати усереднюються для отримання загальної оцінки.

2. Leave-One-Out Cross-Validation (LOOCV): LOOCV передбачає навчання моделі на всіх об'єктах, крім одного, і оцінку її продуктивності на залишеному об'єкті. Цей процес повторюється для кожного об'єкта в наборі даних.

3. Stratified Cross-Validation: Ця техніка забезпечує збереження розподілу класів в межах фолдів, що є особливо корисним при роботі з даними з незбалансованим розподілом класів.

4. Time-Series Cross-Validation: Дані часових рядів вимагають особливих розглядів через їх часовий характер. Техніки крос-валідації для часових рядів, такі як впередовий ланцюг або зміщення вікна, використовуються для оцінки моделей оцінювання на послідовних даних.

2.3.3 Порівняння моделей

Порівняння різних моделей оцінювання є невід'ємною частиною вибору найбільш відповідної моделі для конкретного контексту інженерії програмного забезпечення. При порівнянні моделей слід враховувати кілька факторів, зокрема:

1. Метрики продуктивності: Оцінюйте та порівнюйте моделі за допомогою відповідних метрик ефективності, таких як MAE, RMSE або R^2, щоб визначити, яка модель забезпечує найкращу точність та точність оцінювання.

2. Специфічність домену: Враховуйте конкретні вимоги та характеристики галузі інженерії програмного забезпечення. Деякі моделі можуть показувати кращі результати в певних контекстах, таких як гнучка розробка чи проекти великого масштабу.

3. Доступність даних: Оцініть доступність та якість даних, необхідних для кожної моделі. Деякі моделі можуть мати вищі вимоги до даних або працювати краще з великими наборами даних [65].

4. Складність моделі: Розгляньте складність та інтерпретованість моделей. Простіші моделі можуть бути обрані, коли ключовими є прозорість та легкість розуміння.

5. Стійкість: Оцініть стійкість моделей, вивчаючи їх продуктивність в різних сценаріях та наборах даних. Стійкі моделі повинні послідовно забезпечувати точні оцінки та прогнози, незалежно від варіацій у вхідних даних.

6. Масштабованість: Розгляньте масштабованість моделей, особливо при роботі з проектами програмного забезпечення великого масштабу. Бажано, щоб моделі ефективно впоралися з зростаючим обсягом даних та складністю.

7. Обмеження та припущення: Оцініть обмеження та припущення кожної моделі. Деякі моделі можуть робити певні припущення, які можуть не відповідати конкретним контекстам інженерії програмного забезпечення. Розуміння цих обмежень є важливим для прийняття обґрунтованих рішень.

8. Практичні врахування: Беріть до уваги практичні аспекти, такі як складність впровадження, обчислювальні вимоги та легкість інтеграції в існуючі процеси розробки програмного забезпечення. Моделі, які легко впроваджувати, утримувати та впроваджувати в існуючі робочі процеси, часто є бажаними.

2.3.4 Візуалізація та інтерпретація

Техніки візуалізації можуть сприяти оцінці та порівнянню моделей оцінювання. Візуалізація оцінених значень поряд з реальними значеннями може надати чітке уявлення про продуктивність моделі. Графічні представлення, такі як точкові діаграми, лінійні графіки або стовпчасті діаграми, можуть допомогти виявити закономірності, тенденції, викиди та будь-які розходження між оціненими та реальними значеннями (66).

Крім того, інтерпретованість є важливим аспектом при порівнянні моделей. Моделі, які надають зрозумілі результати, такі як зрозумілі рівняння або рейтинги

важливості ознак, дозволяють зацікавленим особам отримати відомості про фактори, що впливають на оцінки та прогнози. Ця прозорість підвищує довіру до моделей та сприяє процесам прийняття рішень.

2.4 Кращі практики та рекомендації

У аналізі існуючих моделей, методів та алгоритмів для оцінок та прогнозів у програмному інженерії важливо враховувати найкращі практики та рекомендації, які можуть підвищити ефективність та надійність процесів оцінювання. У цьому розділі розглядаються деякі ключові найкращі практики та надаються рекомендації для досягнення точних та надійних оцінок у програмному інженерії.

2.4.1 Підготовка даних та їх якість

Одним із фундаментальних аспектів оцінки є підготовка даних. Правильний збір, очищення та попередня обробка даних є важливими для точних оцінок. Розглянемо наступні найкращі практики:

1. Збір даних: Збирайте дані з надійних джерел, які є представниками контексту програмного інженерії, який розглядається. Забезпечте, щоб дані охоплювали широкий спектр характеристик проекту, таких як обсяг, складність та методологія розробки.

2. Очищення даних: Вилучайте викиди, дублікати або неактуальні точки даних, які можуть негативно впливати на моделі оцінок. Проводьте ретельні процеси очищення даних для забезпечення якості та цілісності набору даних.

3. Вибір ознак: Визначайте та обирайте найбільш інформативні ознаки для оцінки. Уникайте включення зайвих або висококорельованих ознак, які можуть вносити шум чи відхилення в моделі.

4. Нормалізація даних: Нормалізуйте дані до однорідного масштабу, щоб усунути будь-які відхилення, спричинені різницею в одиницях вимірювання або

діапазонах. Звичайні техніки нормалізації включають масштабування min-max або z-оцінки.

5. Обробка пропущених даних: Розробляйте стратегії обробки відсутніх даних, такі як техніки заповнення пропущених значень або розгляд можливості використання відповідних піднаборів даних з мінімальною кількістю відсутніх значень.

2.4.2 Вибір моделі та валідація

Вибір відповідної моделі оцінювання є важливим для точних передбачень. Розглянемо наступні рекомендації:

1. Розуміння припущень моделі: Ретельно розумійте припущення, які робить кожна модель оцінювання. Забезпечте, щоб ці припущення відповідали характеристикам та обмеженням контексту програмного інженерії [67].

2. Валідація моделі: Валідуйте обрані моделі за допомогою відповідних технік, таких як крос-валідація або валідація на окремому наборі даних. Оцінюйте їх ефективність як на тренувальних, так і на валідаційних наборах даних для забезпечення загальної узагальненості.

3. Ансамбль-методи: Розгляньте можливість використання ансамблевих методів, таких як комбінування декількох моделей чи використання технік ансамблевого навчання, для використання переваг різних моделей та покращення точності оцінювання.

4. Експертна експертиза в галузі: Залучайте фахівців в галузі до процесів вибору та валідації моделей. Їхні уявлення та знання можуть сприяти вибору моделей, які відповідають конкретним потребам та викликам галузі програмного інженерії.

2.4.3 Постійне вдосконалення та адаптація

Процес оцінювання в програмному інженерії слід розглядати як ітеративний та адаптивний процес. Розгляньте наступні найкращі практики:

1. Постійний моніторинг: Постійно моніторте роботу обраних моделей оцінювання. Регулярно оцінюйте їх точність, точність та надійність, щоб виявити будь-які відхилення або погіршення результатів [68].

2. Зворотний зв'язок та навчання: Впроваджуйте механізми зворотного зв'язку для фіксації та вивчення фактичних результатів програмних проектів. Використовуйте цей зворотний зв'язок для оновлення та вдосконалення моделей оцінювання, враховуючи уроки, вивчені з попередніх проектів.

3. Вимірювання та порівняння: Порівнюйте роботу обраних моделей оцінювання зі стандартами галузі або визначеними стандартами, щоб визначити їх ефективність та виявити області для вдосконалення.

4. Співпраця та обмін знанням: Сприяйте співпраці та обміну знанням серед фахівців в області програмного інженерії. Діліться досвідом, найкращими практиками та отриманими уроками для спільного підвищення точності та ефективності практик оцінювання.

5. Прийняття практик Agile: Розгляньте можливість впровадження практик Agile, які наголошують на ітеративному розвитку, частому зворотньому зв'язку та адаптивності. Методології Agile можуть підтримувати постійне вдосконалення процесів оцінювання, дозволяючи вносити корективи та вдосконалення протягом усього життєвого циклу розробки програмного забезпечення.

2.5 Висновки до розділу

У цьому розділі досліджено існуючі моделі, методи та алгоритми для оцінок і прогнозів в області програмного інженерії. Галузь оцінювання програмного забезпечення є критичною для планування проектів, розподілу ресурсів та прийняття рішень. Розуміючи та оцінюючи різноманітні підходи, професіонали в галузі програмного інженерії можуть приймати обґрунтовані рішення для покращення точності та надійності своїх процесів оцінювання.

В розділі оглянуто як традиційні моделі оцінювання, так і гібридні та комбіновані. В результаті проведено оцінку та порівняння цих моделей. Завдяки використанню переваг різних технік оцінювання організації можуть покращити точність та надійність оцінок. Методи ансамблю, ф'южн моделей та інтеграція експертних висновків були визначені як ефективні стратегії для поєднання моделей.

При оцінці та порівнянні моделей оцінювання наголошено на важливості ключових факторів, таких як точність, надійність, масштабованість, обмеження та практичні врахування. Техніки візуалізації та інтерпретації визначено як цінні інструменти для розуміння результатів та продуктивності моделей.

Запропоновано найкращі практики та рекомендації для досягнення точних оцінок. Серед них є підготовка та якість даних, вибір та валідація моделей, а також формування культури постійного вдосконалення та адаптації. Слідуючи цими практиками, фахівці в галузі програмного інженерії можуть покращити свої процеси оцінювання та приймати більш обґрунтовані рішення.

РОЗДІЛ 3
ПРАКТИЧНЕ ЗАСТОСУВАННЯ ПІДХОДУ ОЦІНКИ ТА ПРОГНОЗУВАННЯ ВИКОНУВАНОСТІ ЗАДАЧ ДЛЯ ІТ ПРОЕКТУ

3.1 Анотація та формулювання проблеми

3.1.1 Анотація

Точна оцінка є вирішальною для успіху проектів розробки програмного забезпечення. Однак, постійні проблеми переоцінки та недооцінки продовжують турбувати галузь. Це дослідження розглядає використання методів експертного оцінювання як засобу підвищення точності оцінок завершення ІТ-проектів. Експертне оцінювання передбачає отримання висновків від досвідчених фахівців для визначення зусиль, необхідних для конкретних завдань. Тим не менш, експертне оцінювання вразливе перед різними упередженнями та когнітивними спотвореннями, які можуть підірвати точність оцінок. Це дослідження розглядає основу переоцінки та недооцінки в проектах розробки програмного забезпечення та пропонує практичні засоби для зменшення ризиків, пов'язаних із оцінками на основі експертного оцінювання.

Щоб досягти цього, у дослідженні використовується підхід прикладу і порівнює однобальну оцінку з технікою програмного оцінювання та рецензування (PERT) для поліпшення точності. Результати дослідження показують, що оцінка PERT надає спектр оцінок, включаючи очікуваний випадок, причому середня оцінка більш точно узгоджується з оптимістичним сценарієм, ніж оригінальні оцінки. Крім того, обчислюються інтервали впевненості для надання діапазону оцінок при різних рівнях достовірності. Ці висновки підкреслюють, що оцінка PERT, зі своїм спектром оцінок і розрахунком очікуваного випадку, значно підвищує точність оцінок порівняно з однобальною оцінкою. Крім того, у дослідженні використовується середня відносна похибка (MRE) для вимірювання точності оцінок, демонструючи потенціал для подальших удосконалень у точності оцінок.

Це дослідження збагачує галузь управління IT-проектами, акцентуючи значущість методів експертного оцінювання, вказуючи на труднощі, пов'язані з переоцінкою та недооцінкою, та надаючи практичні інструменти та методології для підвищення точності оцінок. Шляхом обмеження суб'єктивності та сприяння культурі постійного вдосконалення в практиках оцінювання можна виконувати IT-проекти вчасно та в межах бюджету, що призводить до виняткових результатів проекту та підвищення задоволеності зацікавлених сторін.

3.1.2 Формулювання проблеми

Під час створення комп'ютерних програм дуже важливо уявляти, скільки часу це займе, скільки коштуватиме і що нам потрібно, щоб завершити створення цієї програми [69-70]. Та навіть якщо у нас є багато способів та інструментів щоб “вгадати” ці дані, ми все одно часто здогадуємось занадто багато або занадто мало. Це викликає проблеми.

Таким чином, нам потрібно звернути увагу на наступне:

1. Експертне оцінювання [71-72]: Це означає запитувати думку досвідчених людей, коли ми вгадуємо.

2. Чому ми помиляємося: Ми повинні розуміти, чому ми робимо ці помилкові вгадування.

3. Поради, як здогадуватись правильно: Ми також повинні ділитися корисними ідеями того, як робити кращі здогадки у комп'ютерних проектах.

Роблячи це, ми можемо покращити управління комп'ютерними проектами. Ми будемо краще здогадуватись, і це допоможе нам завершити роботу вчасно і в межах нашого бюджету. Таким чином, всі, хто залучений до проекту, будуть задоволені результатами.

Коли ми помиляємося щодо того скільки часу займе проект, це може викликати багато проблем. Ці проблеми включають те, що проект займає більше часу, коштуватиме більше грошей, ніж планувалося, і не матимете задуманий функціонал. Це робить людей, які залучені до проекту, невдоволеними і може завдати шкоди бізнесу.

Один із способів краще вгадувати - це запитувати думку досвідчених людей. Це називається експертним оцінюванням. Це важливо в комп'ютерних проектах, оскільки нам часто доводиться здогадуватись, наскільки проект дуже складний та нестабільним.

Таким чином, дуже важливо розуміти, чому ми робимо ці помилкові вгадування і вивчати, як їх уникнути. Це може зробити комп'ютерні проекти краще. Вони будуть завершені вчасно і не будуть коштувати більше, ніж планувалося.

3.2 Техніка експертного оцінювання

Експертне оцінювання - широко використовуваний підхід для оцінки того, скільки часу і зусиль знадобиться для завершення проектних завдань. Цей метод передбачає звернення до досвідчених фахівців, які мають глибоке розуміння роботи, і вони надають свої висновки для створення обґрунтованого припущення щодо обсягу і складності роботи. Однак, хоча експертне оцінювання може бути цінним інструментом у світі оцінювання програмного забезпечення, воно не позбавлене своєї частки викликів і обмежень, які потребують уважного розгляду.

Експертне оцінювання передбачає практику звертання до одного експерта або групи експертів, щоб отримати їхні думки та висновки про визначенні часу та зусиль, які будуть потрібні для розробки програмного забезпечення. Цей підхід може бути досить ефективним, особливо коли завдання чітко визначено, а експерт або експерти мають відповідний досвід та знання, що стосуються роботи. Давайте глибше зануримося в цей процес та розглянемо, як він може допомогти точно оцінити завдання з розробки програмного забезпечення.

Однак важливо бути обізнаним із кількома обмеженнями, пов'язаними з використанням лише індивідуального експертного судження. Давайте глибше розглянемо ці обмеження, щоб зрозуміти їх значущість:

1. *Вплив упереджень на експертне судження*: По-перше, експерти можуть мати певні упередження, які можуть впливати на їхні оцінки. Ці упередження

можуть виникати на основі їхніх особистого досвіду, вірувань або вподобань. Наприклад, експерт, який має обширний досвід з конкретної технології, схильний переоцінювати час, необхідний для завдань, пов'язаних із цією технологією, оскільки його знайомство може зробити його більш обережним.

2. *Вплив когнітивних упереджень на оцінки*: По-друге, експерти не є відділені від когнітивних упереджень, які можуть призводити як до переоцінки, так і до недооцінки завдань. Одним із таких упереджень є ефект якору, де початкова оцінка, часто надана колегами або зовнішніми джерелами, може сильно впливати на наступні оцінки. Це означає, що, якщо експерт спочатку чує високу оцінку, він може підсвідомо коригувати свою власну оцінку вверх, навіть якщо це може бути необґрунтовано.

3. *Обмежена експертиза в конкретних областях*: По-третє, експерти можуть не мати комплексних знань або досвіду в усіх областях проекту. Відповідно до цього їхні оцінки можуть бути неповними у випадку завдань, які вимагають спеціалізованого досвіду, якого вони не мають. Це може призвести до недооцінки, оскільки вони можуть недооцінювати складнощі, пов'язані з цими конкретними завданнями.

4. *Групова динаміка та консенсус*: Додатково, у випадках, коли в процесі оцінки бере участь група експертів, може виникнути тенденція підкорятися груповому консенсусу. Ця конформність може пригнічувати різноманітні точки зору і призводити до оцінок, які не є настільки точними, як це може бути.

5. *Самовпевненість*: Деякі експерти можуть виявляти самовпевненість у своїх судженнях, вважаючи свої оцінки більш точними, ніж вони є насправді. Ця самовпевненість може вести до неправильних суджень і наступних проблем з проектом.

6. *Зміна обставин*: Контекст проекту може змінюватися з часом, вводячи нові змінні та виклики. Початкова оцінка експерта може не враховувати цих змінюючихся факторів, що призводить до неточностей по мірі розвитку проекту.

Враховуючи ці обмеження, в той час як індивідуальне експертне оцінювання є цінним інструментом, його слід доповнювати іншими техніками оцінювання та

уважністю до потенційних упереджень і прогалин у знаннях. Цей комплексний підхід може підвищити точність та надійність оцінок проекту, в кінцевому підсумку сприяючи більш успішним результатам проекту.

3.3 Приклад проекту

Давайте розглянемо приклад, щоб краще зрозуміти проблему: уявіть команду, якій доручено оцінити, скільки часу знадобиться для додавання абсолютно нової функціональності в існуючий продукт. Це стабільний продукт, що розвивається протягом двадцяти років і має багату історію. Проблемою є те, що команда розробки нової функціональності тільки нещодавно стала частиною проекту, приєднавшись лише протягом останніх шести місяців.

Ця ситуація створює певну суперечливість. З одного боку, у команди є певний досвід і інформація про проект завдяки їхній участі впродовж останніх півроку. Але, коли врахувати обширний часовий проміжок понад 20 років, їхні знання охоплюють лише невеликий відрізок всього проекту. Більшість тонкостей та особливостей проекту є поза їхн полем зору.

Щоб зробити приклад ще складнішим, команді поставлено чіткий термін виконання завдання, який не можна змінити. У команди є всього один день, щоб надати свої оцінки. Це залишає їх з дуже обмеженим простором для ретельного вивчення та аналізу всіх вимог проекту, конструктивних урахувань та інших важливих деталей. Це подібно до того, як вам вручають складний пазл, в якому бракує кількох частин, і від вас очікується його скласти за рекордно короткий час. Завдання - це не просто оцінка виконання проекту, це оцінка в умовах невизначеності та обмежених ресурсів, що робить його справжнім випробуванням їхньої експертизи та адаптивності.

3.4 Використання техніки однобального оцінювання для завдання

Спершу була використана техніка однобальної оцінки. Техніка однобальної оцінки - це простий метод, який використовується в управлінні проектами та розробці програмного забезпечення для передбачення тривалості завдання, вартості чи кількості ресурсів, які воно вимагатиме. У цій техніці для кожного завдання або активності надається одне значення або оцінка без урахування можливих варіацій чи невизначеностей.

У своїй основі, це схоже на внесення одного прогнозу чи передбачення щодо тривалості, вартості чи потреб у ресурсах для завдання. Наприклад, якщо ви оцінюєте, скільки часу знадобиться на розробку конкретної функціональності програмного забезпечення, ви скоріш за все просто скажете щось на кшталт: "*Я думаю, що це займе 5 днів*". Це одне число представляє ваш найкращий прогноз на основі вашого поточного знання та досвіду, але воно не враховує можливих ризиків, несподіваних затримок чи варіацій, які можуть виникнути під час виконання завдання.

Однобальна оцінка досить швидка та проста у використанні, що робить її доречною для початкового планування проекту або в тих випадках, коли обмежена деталізована інформація.

Простими словами, експертне оцінювання означає збір експертів і отримання від них приблизного висновку про те, як довго чи складно буде щось зробити, наприклад, додати нову функціональність. І саме так відбулося в цій ситуації.

Ефективною стратегією для отримання більш точної оцінки того, скільки часу знадобиться для завершення нової функціональності, є розбиття її на менші, більш керовані частини. Цей підхід був впроваджений у цьому конкретному випадку. Ось як це працювало:

Команда дотримувалися принципу, згідно з яким кожна з цих менших частин, які називали "частинами", повинна бути досяжною протягом двох тижнів або менше. Крім того, ці частини повинні були виконуватися одним розробником.

У команди є чітке правило коли саме позна визнати функціональність “виконаною”. Спочатку розробнику потрібно написати код для функціональності. Потім її повинні переглянути інші колеги, щоб переконатися, що вона відповідає вимогам проекту та стандартам якості. Після цього функціональність слід інтегрувати з іншими частинами проекту. Ця фаза інтеграції гарантувала, що вона працює безперешкодно з іншими компонентами. Наступною є ретельна перевірка для виявлення та виправлення всіх можливих проблем чи помилок. Тільки після успішного завершення всіх цих кроків функціональність можна вважати "виконаною".

Відповідно до цього систематичного підходу до розбиття функціональності на менші, керовані частини та забезпечення того, щоб кожна частина виконувалася протягом двох тижнів одним розробником, метою команди стало покращення точності оцінки та планування проекту. Цей метод дозволяє краще контролювати та вимірювати прогрес, що сприяє більш ефективному управлінню проектом та прийняттю рішень. Результати оцінки представлені в таблиці 3.1.

Таблиця 3.1

Оцінювання проекту за допомогою однобальної оцінки

Частина	**Кількість днів на виконання**
Частина 1	6
Частина 2	4
Частина 3	6
Частина 4	10
Частина 5	6
Частина 6	6
Частина 7	5
Разом	**43**

Зазвичай процес формування оцінок закінчується на цьому етапі. Проте деякі керівники проектів люблять бути особливо обережними, тому вони додають певну додаткову кількість днів, зазвичай приблизно 30% до існуючої цифри, як свого роду "страховий запас". У цьому конкретному випадку це підвищить загальну оцінку до 56 днів.

Коли ці оцінки представляються людям, які цікавляться проектом, таким як зацікавлені особи (стейкхолдери), у них часто виникає багато запитань. Вони хочуть знати точно, як ми прийшли до цих чисел. Таким чином, коли вони виявляють, що початкова оцінка становить 43 дні, а менеджер вирішив додати ще 13 днів як запас, відбувається цікавий момент. Люди зазвичай фокусуються на початковій оцінці в 43 дні як на реальне зобов'язання, і вони просто ігнорують більше число (56 днів), оскільки вони бачать його як додатковий запобіжник. Це означає, що оцінка в 43 дні стає для них більш важливою та обов’язковою.

3.5 Практичне використання техніки PERT для завдання

Однобальна оцінка є досить прямолінійною і не надає ніякої гнучкості або діапазону можливостей. Щоб побачити, як це порівнюється з більш вдосконаленим методом, ми звернулися до Програмної методики оцінювання та рецензування (Program Evaluation and Review Technique), або коротко PERT [73-75].

PERT - це метод, який використовується в управлінні проектами та оцінці завдань. Він призначений для того, щоб допомогти керівникам проектів робити більш обґрунтовані передбачення щодо того, як довго займе завершення проекту. PERT особливо корисний для проектів, які мають багато невизначеностей та змінних.

За допомогою PERT можна зробити щось дійсно цікаве. Замість того, щоб мати лише одну оцінку, ми можемо розрахувати дещо під назвою “Expected Case” (Очікуваний Випадок). Цей Expected Case не завжди розташований точно посередині

між найкращим і найгіршим сценаріями; він може бути десь інде, даючи нам більш детальний погляд на те, що може відбувається.

Після короткої перерви експертам було представлено нове завдання. Їх нова задача полягала в тому, щоб надати оцінки для двох відмінних сценаріїв - одного, який представляв би найкращий результат, і іншого, що зображував би найгірший.

Давайте глибше вникнемо в ці сценапрії:

1. *Best Case Scenario (найкращий сценарій)*: У цьому сценарії експертам було запропоновано уявити ситуацію, де все розгортається дивовижно гладко і бездоганно. Це схоже на створення картини ідеального проектного шляху, де немає непорозумінь, неочікуваних проблем і затримок. Все відбувається як за годинниковим механізмом, і завдання виконується в рекордно короткий час.

2. *Worst Case Scenario (найгірший сценарій)*: З іншого боку, експертам доручено уявити сценарій, де все, що може піти не так, дійсно йде не так. Це схоже на витвір найбільш складного та хаотичного проектного середовища, яке можна уявити. Затримки, проблеми та перешкоди на кожному кроці створюють надзвичайно невигідну ситуацію, де виконання завдання займає набагато більше часу, ніж очікувалося.

Досліджуючи обидва сценарії, від найкращого до найгіршого, експертам рекомендували розглядати широкий спектр можливостей та викликів. Цей всебічний підхід допомагає краще розуміти потенційну змінність та ризики, пов'язані з завданням, в кінцевому підсумку призводячи до більш жорсткого планування та управління проектом. Результати оцінювання наведені на рисунку 3.1.

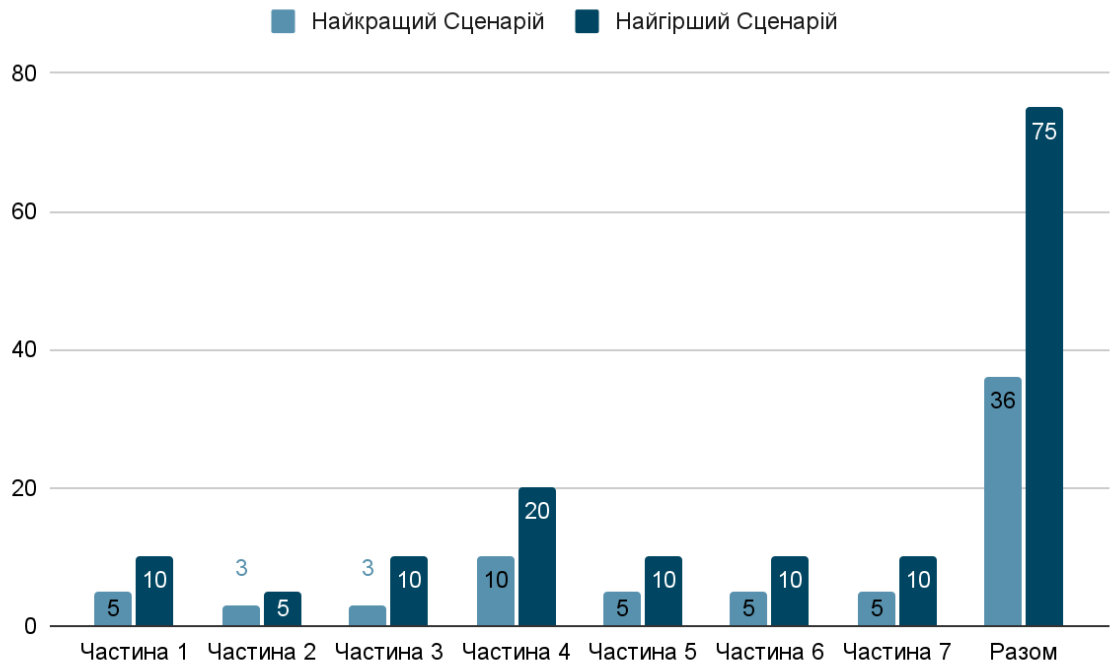

Рис. 3.1. Оцінювання проекту за допомогою техніки PERT

Давайте глибше розглянемо, чим різняться ці оцінки. Порівнюючи початкову однобальну оцінку в 43 дні з оцінками як для найкращого, так і для найгіршого сценарію, ми можемо здобути цікаві висновки.

Початкова оцінка, яка була 43 дні, здається більш близькою до оцінку з найкращого сценарію. У найкращому сценарії все йде надзвичайно гладко, і відзначається, що завдання займе всього 36 днів. Це не дуже далеко від початкової оцінки.

Але, порівнявши це з оцінкою з найгіршого сценарію, яка передбачає ситуацію, де все йде не так, і завдання завершується страшними 75 днями, початкова оцінка виглядає значно меншою.

По своїй суті, це говорить нам про те, що початкова оцінка більше схиляється до оптимістичного сценарію, де все йде надзвичайно добре, ніж до сценарію, де все йде погано.

Ми вже визначили важливість визначення як найкращого сценарію, так і найгіршого. Однак, наявність декількох сценаріїв надає нам досить широкий спектр

можливих оцінок, залишаючи нас перед важливим питанням: "На яку оцінку ми повинні опиратися?"

Жодна з оцінок, які ми обговорювали до цього, включаючи початкові, і навіть середнє значення, не є ідеальним вибором. Чому, ви можете запитати?

Справа в тому, що найгірший сценарій часто виявляється набагато гіршим, ніж ми зазвичай очікуємо середньому сценарії, який ми називаємо "Очікуваний сценарій" (Expected Case). Це схоже на широкий спектр, з найгіршим сценарієм на одній стороні та очікуваним сценарієм десь посередині. Якщо ми просто візьмемо середнє значення всіх цих варіантів, ми можемо отримати оцінку, яка є занадто високою.

Ця вища оцінка може викликати різноманітні труднощі в плануванні проекту та прийнятті рішень. Ключова проблема тут полягає в тому, що знайти правильну оцінку не так просто, як вибрати один із сценаріїв чи усереднити їх усіх. Це передбачає більш витончений підхід, який враховує ймовірності та спектр можливостей для прийняття обґрунтованих та збалансованих рішень в управлінні проектом.

Для правильного використання методу PERT додається додаткова оцінка для Найбільш Ймовірного Сценарію (Most Likely Case), яка також визначається за допомогою експертного оцінювання.

Результати після додавання оцінки для найбільш ймовірного сценарію подано на рисунку 3.2.

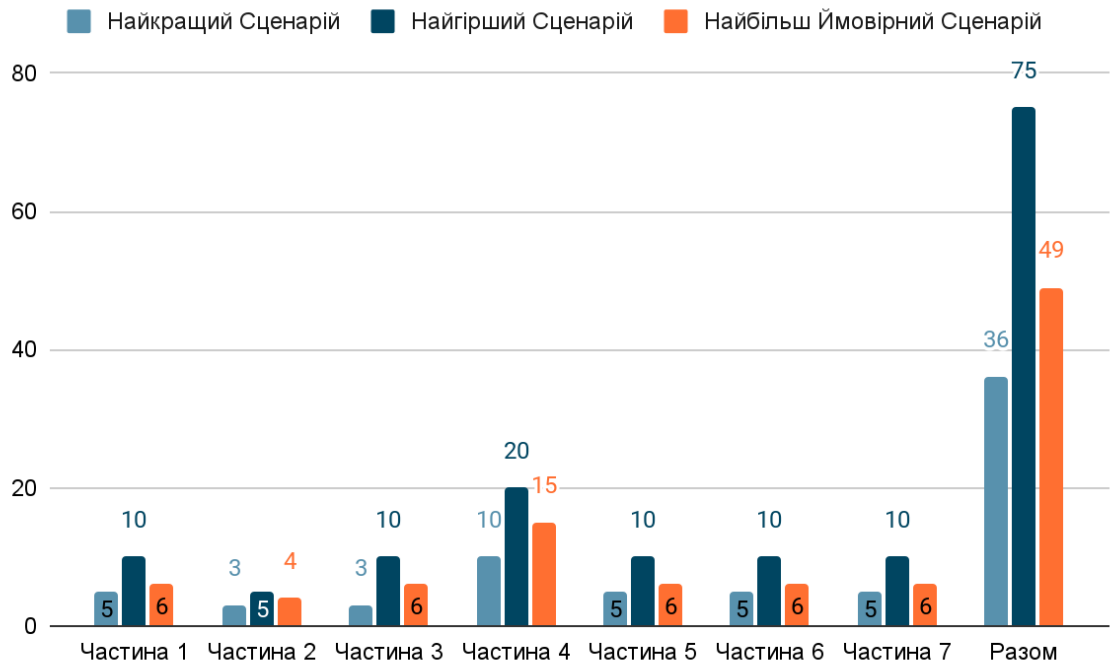

Рис. 3.2. Оцінювання проекту за допомогою техніки PERT із додаванням найбільш ймовірного сценарію

В результаті найбільш ймовірний сценарій все ще ближчий до найкращого в 36 днів, але фактично довший, ніж початкова оцінка в 43 дні.

Згідно з методом PERT, ми можемо використовувати наступну формулу для розрахунку Очікуваного Сценарію (Expected Case):

Expected Case = (Best Case + (4 x Most Likely Case) + Worst Case) / 6

Очікуваний Сценарій = (Найкращий + (4 x Найбільш Ймовірний) + Найгірший) / 6

Ця формула враховує як широкий спектр всього діапазону, так і те, де знаходиться найбільш ймовірний сценарій всередині цього діапазону.

Рисунок 3.3 представляє результати з попередньої таблиці з додаванням Очікуваного Сценарію.

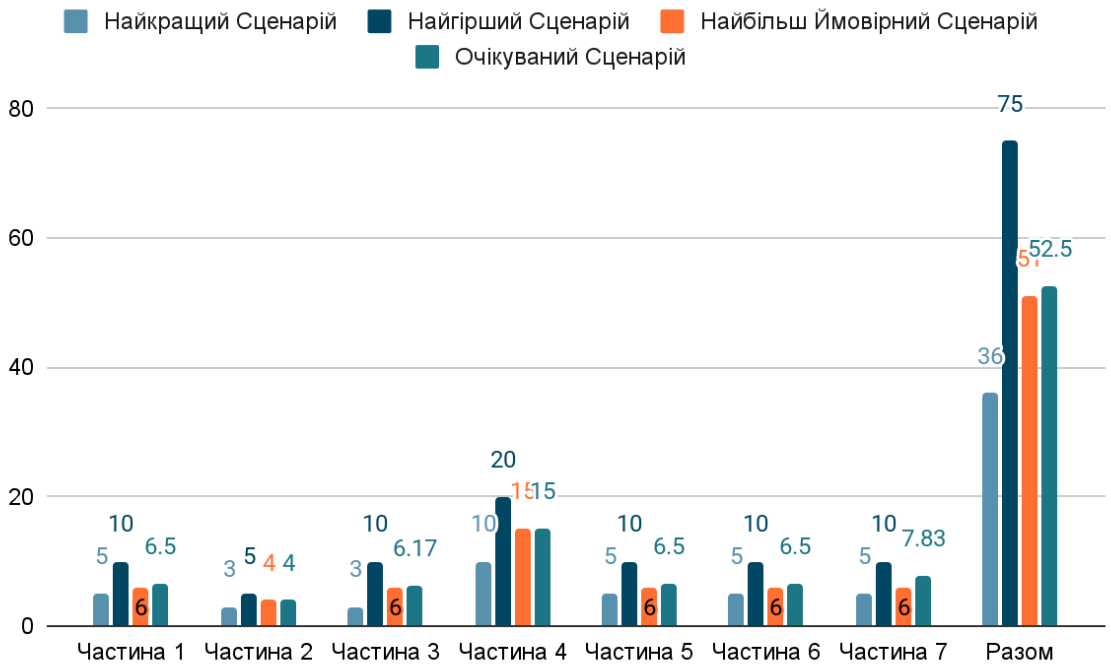

Рис. 3.3. Оцінювання проекту за допомогою техніки PERT із додаванням найбільш очікуваного сценарію

В результаті Очікувана Оцінка становить 52,50 днів.

Хоча результати, отримані за допомогою методики оцінювання PERT, пропонують більше, ніж одне число, вони все ще є числовими значеннями, які не мають рівня достовірності.

Для цього нам потрібно розрахувати діапазон за допомогою наступних формул на основі нормального розподілу:

Lower Bound = Expected Case - (Range * Z)

Upper Bound = Expected Case + (Range * Z)

Нижня Межа = Очікуваний Сценарій - (Діапазон * Z)

Верхня Межа = Очікуваний Сценарій + (Діапазон * Z)

Z-значення (Z-value), також відоме як Z-оцінка (Z-score) чи стандартний показник, є статистичним показником, використовуваним в контексті стандартного

нормального розподілу. Воно показує, наскільки далеко конкретний показник даних відхиляється від середнього (середнього значення) набору даних, вираженого в термінах стандартних відхилень.

Ось що означає Z-значення:

1. *Віддаленість від Середнього*: Z-значення вимірює, на скільки стандартних відхилень даний показник даних відхиляється від середнього. Якщо Z-значення дорівнює нулю, це означає, що показник даних розташований точно на середньому рівні. Якщо воно позитивне, то воно вище середнього, і якщо воно від'ємне, то нижче середнього.

2. *Стандартизація*: Z-значення використовується для стандартизації даних, що полегшує порівняння різних наборів даних, які можуть мати різні одиниці чи шкали. Перетворивши дані в Z-значення, ви можете оцінити, наскільки несподіваним або типовим є певний показник даних у межах свого набору даних.

3. *Інтерпретація*: Зазвичай в стандартному нормальному розподілі приблизно 68% даних знаходиться в межах одного стандартного відхилення від середнього (між -1 і +1 Z-значеннями), приблизно 95% — в межах двох стандартних відхилень (-2 до +2 Z-значення), і приблизно 99,7% — в межах трьох стандартних відхилень (-3 до +3 Z-значення).

Z-значення є корисними в різних галузях, таких як статистика, фінанси і контроль якості, оскільки вони дозволяють порівнювати точки даних з різних наборів даних і розуміти відносне положення та значущість кожної точки даних у межах свого власного набору даних. Вони забезпечують стандартизований спосіб оцінки того, наскільки екстремальним чи загальним є спостереження в даному контексті.

На основі цих даних можна розрахувати нижню та верхню межі для кожного Рівня Впевненості (Confidence Level). Вони будуть представляти діапазон оцінок.

Таблиця 3.2

Обчислення верхньої та нижньої меж на основі нормального розподілу

Рівень Впевненості	**Нижня Межа**	**Верхня Межа**
80%	52.5 - (6.5 * 1.282) ≈ 44,17	52.5 + (6.5 * 1.282) ≈ 60,83
85%	52.5 - (6.5 * 1.440) ≈ 43,14	52.5 + (6.5 * 1.440) ≈ 61,86
90%	52.5 - (6.5 * 1.645) ≈ 41,81	52.5 + (6.5 * 1.645) ≈ 63,19
95%	52.5 - (6.5 * 1.960) ≈ 39,76	52.5 + (6.5 * 1.960) ≈ 65,24
99%	52.5 - (6.5 * 2.576) ≈ 35,76	52.5 + (6.5 * 2.576) ≈ 69,24
99.9%	52.5 - (6.5 * 3.291) ≈ 31,11	52.5 + (6.5 * 3.291) ≈ 73,89

В сфері управління проектами в галузі IT, коли мова йде про рівні впевненості, все, що нижче 90%, зазвичай вважається занадто великим ризиком. Навпаки, оцінки вище цього порогу, як правило, мають дуже широкий спектр можливостей.

Для прикладу, з рівнем довіри 90%, очікувані зусилля для проекту можуть варіюватися від 42 до 63 днів. Це все ще досить значний діапазон, що становить 50% різниці між найкоротшим і найдовшим можливим терміном. Не рідко багато зацікавлених сторін приділяють більше уваги нижньому кінцю цього діапазону, ігноруючи вищий показник.

Проте важливо відзначити, що надання докладного пояснення того, як були отримані ці цифри, може значно покращити розуміння і прийняття рішень. Це висвітлює фактори та обставини, які враховувалися в процесі оцінки, допомагаючи зацікавленим сторонам усвідомити повну картину і приймати більш обґрунтовані рішення щодо планування та управління проектом.

3.6 Порівняння прогнозованих оцінок з фактичними результатами

Давайте уважніше розглянемо цифри, які ми представили зацікавленим сторонам проекту. Ці числа мають вагоме значення у наших дискусіях з управління проектом. Є два ключові показники:

1. *Expected Case Estimate (Оцінка Очікуваного Сценарію)*: Ця оцінка складає близько 52,50 дня. Вона представляє найкраще обгрунтоване припущення, засноване на наявній інформації, про те, скільки часу може знадобитися на завершення проекту. Це як наше проміжне припущення.

2. *Range of Possibilities (Діапазон Можливостей)*: Ми також надали діапазон, який охоплює період від 42 до 63 днів. Цей діапазон особливо важливий, оскільки він включає потенційні результати з рівнем впевненості у 90%. Іншими словами, ми вважаємо, що існує велика ймовірність того, що фактична тривалість буде входити в цей діапазон.

Тепер ці дані безсумнівно цінні для управління нашим проектним плануванням та прийняття рішень. Однак, вони набувають ще більшої ваги, коли ми порівнюємо їх із "Фактичними Результатами" (Actual Results). Цей термін вказує на те, що відбулося під час проекту - реальний час, який витратили на виконання.

Бачите, саме порівняння наших обдуманих оцінок із фактичними результатами надає нам найбільш значущу та практичну інформацію. Це дозволяє нам оцінити точність та надійність наших прогнозів, дозволяючи налаштовувати наш підхід та приймати більш обґрунтовані рішення в майбутньому. Таким чином, хоча оцінки надають нам міцний фундамент, саме порівняння з фактичними результатами справжньою мірою підтверджує та вдосконалює наші стратегії управління проектом.

Також потрібно обрахувати Mean Relative Error (MRE) за допомогою наступної формули:

MRE = |(Actual Result - Expected Case) / Actual Result|

MRE = |(Фактичний Результат - Очікуваний Сценарій) / Фактичний Результат|

Таблиця 3.3 представляє очікувані та фактичні результати, а також середні відносні похибки (MRE) для всіх частин.

Таблиця 3.3

Порівняння очікуваних і фактичних результатів

Частина	**Кількість днів на виконання**		
	Очікуваний Сценарій	**Фактичні результати**	**MRE**
Частина 1	6.5	6	8%
Частина 2	4.0	2	100%
Частина 3	6.17	6	3%
Частина 4	15	20	25%
Частина 5	6.5	10	35%
Частина 6	6.5	8	19%
Частина 7	7.83	7	12%
Разом	**52.50**	**59**	
Середнє значення MRE для оцінок частин			**29%**

3.7 Висновки до розділу

У розділі описано реальний проект та використання на ньому певних технік оцінювання виконання конкретної задачі. Використовувались техніки експертного оцінювання, такі як: техніка однобального оцінювання, техніки PERT з використанням найкращого та найгіршого сценаріїв, додано варіант з найбільш ймовірним сценарієм.

За результатом усіх даних розраховано очікуваний сценарій та проаналізовано його значення у порівнянні з найкращим, найгіршим та найбільш ймовірним сценаріями.Використано нормальний розподіл для обрахування похибки.

І, що є найголовнішим, проведено порівняння оцінок з вже фактичними результатами. В результаті цього порівняння виведено похибку Mean Relative Error (MRE), яка використовувалась командою в подальших оцінках.

ВИСНОВКИ

В даній магістерській роботі було розглянуто проблему оцінки та прогнозування виконуваності задач в програмній інженерії. Як з’ясувалось, проблема існувала і буде існувати завжди, тому потрібно збирати метрики та займатись постійним вдосконаленням вимірювання оцінки на постійній основі.

Було проаналізовано основні підходи до рішення проблеми оцінки та прогнозування виконуваності задач в програмній інженерії та досліджено існуючі моделі. Найпопулярнішим методом оцінки виявився метод експертного оцінювання. Значна частина оцінювання проектів, від 70% до 90%, ґрунтується на експертній думці та висновках спеціалістів.

Саме метод експертного оцінювання використано як основний для оцінки виконання конкретного завдання на реальному проекті. Використовується метод, який порівнює однозначну оцінку з технікою програмного оцінювання та рецензування (PERT) з метою покращення точності. Результати розгляду показують, що оцінка PERT надає широкий спектр оцінок, включаючи очікуваний випадок, і середні оцінки узгоджуються точніше з оптимістичним сценарієм, ніж оригінальні оцінки.

Крім того, обчислені інтервали впевненості надають діапазон оцінок при різних рівнях достовірності. Ці висновки підкреслюють, що оцінка PERT, з своїм різноманіттям оцінок і розрахунком очікуваного випадку, значно підвищує точність оцінок порівняно з однозначною оцінкою.

Додатково, середній відносний розмах похибки (MRE) використовується для вимірювання точності оцінок, що демонструє потенціал для подальших поліпшень у точності оцінок. Надалі MRE повинно допомагати оцінювати прогрес і точність процесів оцінювання. Це спосіб забезпечення того, що оцінки стають більш надійними, що в кінцевому підсумку призводить до кращого планування та управління проектами.

Ідеєю роботи є бажання проілюструвати перетворення оцінювання з дещо суб'єктивного в структуровану та розвиваючу практику. Таким чином, з плином часу,

оцінки мають стати більш надійними та краще відповідати фактичним результатам проектів [76].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Software Development Cost Estimation: Traditional Versus Contemporary Approaches”, Deek, F. P., & McHugh, J. A. (Information and Software Technology, 2018, pp. 95-113)
2. “50 years of software engineering”, H Erdogmus, N Medvidović, F Paulisch (IEEE Software, 2018)
3. “A software engineering perspective on engineering machine learning systems: State of the art and challenges”, G Giray (Journal of Systems and Software, 2021)
4. “Software simplified”, A Silver (Nature, 2017)
5. “The history of software engineering”, G Booch (IEEE Software, 2018)
6. “Soft sides of software”, LF Capretz, F Ahmed, FQB da Silva (Information and Software Technology, 2017)
7. “Project Management in Construction (7th Edition)”, Sidney M. Levy (McGraw-Hill Education, 2018)
8. “Construction project management: an integrated approach (10th Edition)”, JR Meredith, SM Shafer, SJ Mantel Jr (Wiley, 2017)
9. “Advanced Project Management (4th Edition)”, Frederick Harrison and Dennis Lock (Routledge, 2017)
10. “Project management: a systems approach to planning, scheduling, and controlling (12th Edition)”, H Kerzner (Wiley, 2017)
11. “Information Technology Project Management”, Schwalbe, K (Cengage Learning, 2018)
12. “Project Management for Engineering, Business and Technology (6th Edition)”, John M. Nicholas, Herman Steyn (Routledge, 2020)
13. “Optimizing the power of action learning (3rd Edition)”, MJ Marquardt, S Banks, P Cauwelier (Nicholas Brealey America, 2018)
14. “Cross-boundary teaming for innovation: Integrating research on teams and knowledge in organizations”, AC Edmondson, JF Harvey (Human Resource Management Review, Elsevier, 2018)

15. “Enhancing employee creativity via individual skill development and team knowledge sharing: Influences of dual-focused transformational leadership”, Yuntao Dong, Kathryn M. Bartol, Zhi-Xue Zhang, Chenwei Li (Journal of Organizational Behavior, Wiley Online Library, 2017)
16. “Teacher collaboration in curriculum design teams: effects, mechanisms, and conditions”, Joke M. Voogt, Jules M. Pieters, Adam Handelzalts (Teacher Learning Through Teacher Teams, 2018)
17. “The factors influencing the success of on-going agile software development projects”, C Tam, EJ da Costa Moura, T Oliveira (International Journal of Project Management, 2020, pp. 165-176)
18. “The Rise and Evolution of Agile Software Development”, Rashina Hoda, Norsaremah Salleh, John Grundy (IEEE, 2020)
19. “Proactive risk management: Controlling uncertainty in product development”, PG Smith, GM Merritt (CRC Press Taylor & Francis Group, 2020)
20. “Managing project uncertainty”, David Cleden (Routledge, 2017)
21. “Managing risk in projects”, David Hillson (Routledge, 2017)
22. “Principles of risk analysis: decision making under uncertainty (2nd Edition)” (CRC Press, 2019)
23. “Fundamentals of risk management: understanding, evaluating and implementing effective risk management (5th Edition)”, P Hopkin (Kogan Page, 2018)
24. “Project and program risk management a guide to managing project risks and opportunities”, RM Wideman (Project Management Institute, 2022)
25. “Understanding and managing risk attitude (2nd Edition)”, D Hillson, R Murray-Webster (Routledge, 2017)
26. “Engineering risk management”, T Meyer, G Reniers (De Gruyter, 2022)
27. “A literature review on the affective factors that influence data-driven decision-making”, E Sebestyén (Hungarian Educational Research Journal, 2021)
28. “Data driven management in Industry 4.0: a method to measure Data Productivity”, G Miragliotta, A Sianesi, E Convertini, R Distante (IFAC-PapersOnLine, Elsevier, 2018)

29. “On the power of abstention and data-driven decision making for adversarial robustness”, N Balcan, A Blum, D Sharma, H Zhang (ICLR 2021 Conference Blind Submission, 2020)
30. “Using data-driven safety decision-making to realize smart safety management in the era of big data: A theoretical perspective on basic questions and their answers”, B Wang, C Wu, L Huang, L Kang (Journal of Cleaner Production, Elsevier, 2019)
31. “Software Engineering Economics”, Barry W. Boehm (Prentice Hall, 1981)
32. “Software Engineering: A Practitioner's Approach”, Pressman, R. S. (McGraw-Hill Education, 2014)
33. “Theory-W Software Project Management: Principles and Examples”, Boehm, B., & Ross, R. (IEEE Transactions on Software Engineering, 15(7), pp. 902-916, 1989)
34. “Why Assessment of Software Cost Estimates is Biased and What Can Be Done About It”, Kitchenham, B. A., & Mendes, E. (n Software Cost Estimation, Benchmarking, and Risk Assessment, Springer, pp. 142-154)
35. “A Systematic Review of Software Development Cost Estimation Studies”, Jørgensen, M., & Shepperd, M. (IEEE Transactions on Software Engineering, 33(1), pp. 33-53, 2007)
36. “Contemporary Project Management”, Kloppenborg, T. J., & Petrick, J. A. (Cengage Learning, 2015)
37. “Project Planning, Scheduling & Control”, Lewis, J. P. (McGraw-Hill Education, 2011)
38. “Software Engineering”, Sommerville, I. (Pearson, 2016)
39. “Agile software development: The business of innovation”, Highsmith, J., & Cockburn, A. (Computer, pp. 120-127, 2001)
40. “The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game”, Schwaber, K., & Sutherland, J. (Scrum.org, 2017)
41. “Agile Estimating and Planning”, Cohn, M. (Prentice Hall, 2005).
42. “Kanban: Successful Evolutionary Change for Your Technology Business”, Anderson, D. J. (Blue Hole Press, 2010)

43. “The Principles of Product Development Flow: Second Generation Lean Product Development”, Reinertsen, D. G. (Celeritas Publishing, 2009)
44. “Machine Learning in Software Engineering: Where Are We?”, Mendes, E., Mosley, N., Counsell, S., & Lo, D. (Proceedings of the 39th International Conference on Software Engineering (ICSE) Companion (pp. 331-332). IEEE, 2017)
45. “Towards identifying software project clusters with regard to defect prediction”, Jureczko, M., & Madeyski, L. (Proceedings of the 6th International Conference on Predictive Models in Software Engineering (PROMISE), pp. 29-38, 2010)
46. “Use of machine learning techniques for educational purposes: A decision support system for forecasting students' grades”, Kotsiantis, S. B. (Artificial Intelligence Review, 37(4), pp. 331-344, 2012)
47. “An introduction to variable and feature selection”, Guyon, I., & Elisseeff, A. (Journal of Machine Learning Research, 3, pp. 1157-1182, 2003)
48. “Feature selection for knowledge discovery and data mining (Vol. 454)”, Liu, H., & Motoda, H. (Eds.) (Springer Science & Business Media, 2012)
49. “An introduction to statistical learning (Vol. 112)”, James, G., Witten, D., Hastie, T., & Tibshirani, R. (New York: springer, 2013)
50. “A study of cross-validation and bootstrap for accuracy estimation and model selection”, Kohavi, R. (Proceedings of the 14th international joint conference on Artificial intelligence-Volume 2, pp. 1137-1143, 1995)
51. “A review of studies on expert estimation of software development effort”, Jørgensen, M. (Journal of Systems and Software, 84(8), pp. 1290-1304, 2011)
52. “Measuring application development productivity.”, Albrecht, A. J. (Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, pp. 83-92, 1979)
53. “A model for estimating program complexity”, Kitchenham, B. A. (IEEE Transactions on Software Engineering, pp. 242-247, 1985)
54. “Measures for Excellence: Reliable Software on Time, Within Budget”, Putnam, L. H., & Myers, W. (Prentice Hall, 1992)

55. “Introduction to Linear Regression Analysis”, Montgomery, D. C., Peck, E. A., & Vining, G. G. (Wiley, 2012)
56. “Multivariate Data Analysis”, Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (Pearson, 2013)
57. “Introduction to Time Series Analysis and Forecasting”, Brockwell, P. J., & Davis, R. A. (Springer, 2016)
58. “Forecasting: Principles and Practice (2nd Edition)”, Hyndman, R. J., & Athanasopoulos, G. (OTexts, 2018).
59. “Simulation and the Monte Carlo Method (3rd Edition)”, Rubinstein, R. Y., & Kroese, D. P. (Wiley, 2016).
60. “Bayesian Data Analysis (3rd Edition)”, Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (CRC Press, 2013)
61. “Software Economics: A Roadmap”, Boehm, B. W., & Sullivan, K. J. (Proceedings of the 22nd International Conference on Software Engineering (ICSE '00), Limerick, Ireland, 2000)
62. “Top-down induction of decision trees classifiers—a survey”, Rokach, L., & Maimon, O. (IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 35(4), pp.476-487, 2005)
63. “Towards a rigorous science of interpretable machine learning”, Doshi-Velez, F., & Kim, B. (arXiv preprint arXiv:1702.08608, 2017)
64. “Introduction to machine learning”, Alpaydin, E. (MIT press, 2014)
65. “Data quality: Some comments on the NASA software defect datasets”, Shepperd, M., Song, Q., & Sun, Z. (IEEE Transactions on Software Engineering, 39(9), pp. 1208-1215, 2013)
66. “The visual display of quantitative information”, Tufte, E. R. (Graphics Press, 2001)
67. “The elements of statistical learning: data mining, inference, and prediction”, Hastie, T., Tibshirani, R., & Friedman, J. (Springer Science & Business Media, 2009)
68. “Introduction to linear regression analysis (Vol. 821)”, Montgomery, D. C., Peck, E. A., & Vining, G. G. (John Wiley & Sons, 2012)

69. “Software Estimation”, Steve McConnell (Microsoft press, 2016)
70. “Estimating Software-intensive Systems: Projects, Products, And Processes”, Richard D. Stutzke. (Addison-Wesley Professional, 2005)
71. “Forecasts or fortune-telling: When are expert judgements of safety risk valid?”, Rae, A., & Alexander, R. (Safety Science, 99, pp. 156–165, 2017)
72. “Forecasting of software development work effort: Evidence on expert judgement and formal models”, Jørgensen, M. (International Journal of Forecasting, 23(3), pp. 449–462, 2007)
73. “Estimation Of Task Completion Times With The Use Of The PERT Method On The Example Of A Real Construction Project”, Plebankiewicz, E., Juszczyk, M., & Malara, J. (Archives of Civil Engineering, 61(3), pp. 51–62, 2015)
74. “Using Neutrosophic Sets to Obtain PERT Three-Times Estimates in Project Management”, Mohamed, M., Abdel-Basset, M., Hussien, A., & Smarandache, F. (ResearchGate, 2016)
75. “Combination of program evaluation and review technique (PERT) and critical path method (CPM) for project schedule development”, Ba'Its, H. A., Puspita, I. A., & Bay, A. F. (International Journal of Integrated Engineering, 12(3), pp. 68-75, 2020)
76. “Using expert judgement techniques to estimate IT projects completion on time”, Marian Slabinoha, Taras Mykhailov (Information Technology and Society. Issue 2 (8), 2023)